НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Геометричне моделювання в інформаційних системах»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Веб-система з візуального моніторингу якості отримання знань»

Виконав:
студент IV курсу, групи ТР-61
Мітряхін Юрій Віталійович _____

Керівник:
проф. Отрох Сергій Іванович _____

Консультант:
доцент Шалденко Олексій Вікторович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший рівень
Напрям підготовки 122 Комп’ютерні науки та інформаційні технології
Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Мітряхіну Юрію Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Веб система з візуального моніторингу якості отримання знань
Керівник роботи Отрох Сергій Іванович, к.т.н., професор
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи 7 червня 2020 року
3. Вихідні дані до роботи Тимчасове положення про організацію освітнього візуального моніторингу користувачів під час онлайн-навчання
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз існуючих систем моніторингу якості отримання освіти, вибір методів відеофіксації, розробка архітектури системи, вибір та реалізація засобів розробки програмного забезпечення і створення графічного інтерфейсу, розробка програмного продукту для демонстрації роботи системи
5. Перелік ілюстративного матеріалу
Схема архітектури програми, схема MVC, концептуальна модель бази даних, графічні інтерфейси користувача і викладача.

6. Дата видачі завдання ”_13_” _квітня_ 2020 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
Затвердження теми роботи	**13.04.2020**	
Вивчення та аналіз задачі	20.04.2020	
Розробка архітектури та загальної структури системи	29.04.2020	
Розробка структур окремих підсистем	06.05.2020	
Програмна реалізація системи	17.05.2020	
Оформлення пояснювальної записки	02.06.2020	
Захист програмного продукту	**10.06.2020**	
Передзахист	**10.06.2020**	
Захист	17.06.2020	

Студент _____ ____Мітряхін Ю.В._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ ____Отрох С.І._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дипломна робота складається зі вступу, шести розділів, висновку, переліку посилань з 12 найменувань, одного додатку, і містить 22 рисунків, 3. Повний обсяг дипломної дисертації складає 47 сторінок, з яких перелік посилань займає 1 сторінку. Також є 3 додатки які займають 28 сторінок.

Метою роботи було дослідження та удосконалення онлайн навчання, а саме створення системи з візуального моніторингу якості отримання знань. Для реалізації було обрано створення веб сайту, через зручність використання, та можливість доступу з мобільних телефонів, комп'ютерів та інших технічних засобів, у яких є можливість виходу до глобальної мережі Інтернет.

Проведено дослідження сучасних цифрових методів оцінки рівня засвоєння інформації учнями. Проведено порівняльний аналіз існуючих аналогів. Спроектовано програмний продукт. Обрано технології для реалізації системи як веб сайту. Розроблено всі складові програмного забезпечення в якості фінального програмного продукту. Було проведено тестування даної веб-системи та виправлено помилки.

Ключові слова: веб-сайт, користувацький інтерфейс, якість отримання знань, відеофіксація, аналоги, дизайн, нейронні мережі, онлайн навчання.

ABSTRACT

Thesis consists of an introduction, six chapters, conclusion, list of references from 12 titles, one appendix, and contains 22 figures, 3. The full volume of the master's dissertation is 47 pages, of which the list of references is 1 page, appendices - 28 pages.

The aim of the work was to research and improve online learning, namely to create a system for visual monitoring of the quality of knowledge. The implementation was chosen to create a website, for ease of use, and possible access from mobile phones, computers and other technical means that have the ability to access the global Internet.

A study of modern digital methods for assessing the level of information assimilation by students. A comparative analysis of existing analogues. The software product is designed. Selected technologies for the implementation of the system as a website. All components of the software have been developed as a final software product. This web system has been tested and bugs have been fixed.

Keywords: website, user interface, quality of knowledge acquisition, video recording, analogues, design, neural networks, online learning.

ЗМІСТ

Перелік умовних позначень 8

Вступ 9

1. Постановка задачі 10

2. Аналіз існуючих систем моніторингу якості отримання знань 12

2.1 Проект в сфері масової онлайн-освіти – Coursera 12

2.2 Американська платформа онлайн навчання – Udemy 13

3. Моделювання системи 15

3.1 Загальні вимоги 17

3.2 Користувачі системи 17

3.3 Модуль накопичення даних 17

3.4 Проведення освітнього процесу 18

3.5 Моделювання інтерфейсу користувача 18

3.6 Можливості модернізації 19

3.7 Відеофіксація 19

4. Засоби програмної реалізації 21

4.1 Вибір архітектури програмного комплексу 21

4.2 Опис архітектури серверу 23

4.3 Опис архітектури клієнтського застосунку 25

4.4 Опис інструментів розробки 26

4.5 Обгрунтування вибору програмної реалізації 30

5. Опис програмної реалізації 32

5.1 Опис функціональності системи. 32
5.2 Концептуальна модель бази даних 33
5.3 Опис таблиць бази даних 34
5.4 Модуль кабінету користувача 36
5.4.1 Модуль кабінету викладача 37
5.4.1 Модуль обробки результатів 37
5.5 Комунікація з клієнською частиною 37
6. Методика роботи користувача з програмною системою 39
6.1 Інтерфейс користувача 40
6.2 Інтерфейс викладача 44
Висновки 46
Список використаних джерел 47
Додаток А 48
Додаток Б 50
Додаток В 66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI	–	User Interface
AI	–	Admin Interface
FTL	–	FreeMarker template
MVC	–	Modal View Controller

ВСТУП

Злиття традиційних способів навчання - на основі класної дошки, крейди та монологу перед студентами з сучасними підходами, такими як електронне навчання, орієнтовані на онлайн-мультимедіа додатки і технології, розроблені у співпраці - стали реальною необхідністю для поліпшення навчання. Актуальні інструменти для розробки веб-додатків значно покращилися, і розробники платформ електронного навчання повинні правильно їх використовувати і ефективно розробляти середовища для задоволення потреб вчителів та учнів.

Нові технології дозволяють отримати доступну інформацію для тих, хто зацікавлений в навчанні. За допомогою веб-систем можна вивчати новий матеріал без необхідності залишати свої будинки, за допомогою перегляду цифрових телевізійних програм і / або консультування величезних віртуальних бібліотек курсів, які доступні на відстані в будь-якому куточку світу.

Також важливо не тільки надавати доступ до інформації, а і відслідковувати якість засвоєння опрацьованої інформації. Зазвичай це робиться за допомогою тестів. Основними проблемами навчання онлайн є: низький поріг завершеності курсів учнями з подальшим повторним вивченням навчального матеріалу та надання правильного зворотного відгуку, за яких саме причин були зроблені помилки, наприклад це механічна помилка чи це недосконале вивчення матеріалу.

Зараз набувають популярності технології нейронних мереж. Їхнє використання охоплює багато сфер сучасного життя. Нейронні мережі використовуються в мобільних телефонах, камерах, соціальних мережах, веб сайтах. Особливої популярності набули мережі, які вміють майже безпомилково відслідковувати людей, їхні рухи та емоції.

Тому проаналізувавши це було вирішено зробити крок далі і використати сучасні нейронні мережі для удосконалення онлайн навчання. Мета даної роботи - розробити веб систему, яка за допомогою відеофіксації під час навчання учнів буде надавати результат про якість отримання та давати рекомендації для покращення результатів в майбутньому.

1 ПОСТАНОВКА ЗАДАЧІ

Дистанційне навчання стало популярним з появою інтернету, відкривши нові можливості розвитку для жителів віддалених населених пунктів і ділових людей з щільним робочим графіком. Із швидким розвитком технологій стало зрозуміло, що потенціал онлайн-освіти це щось набагато більше, ніж просто можливість слухати лекції в мережі. Інтернет не тільки зробив навчання доступним широким масам людей (як це сталося у випадку з поширенням університетських курсів), але і повністю змінив сам підхід до процесу навчання.

Велику роль у онлайн навчанні відіграє моніторинг якості освіти. Він найважливіша і невід'ємна частина сучасного повноцінного освітнього процесу.

Якість освіти визначається сукупністю показників, що характеризують різні аспекти навчальної діяльності освітньої установи: зміст освіти, форми і методи навчання, матеріально-технічну базу, кадровий склад, які забезпечують розвиток компетенцій молоді що навчається. Відповідно під моніторингом якості освіти розуміється дослідження всіх цих факторів і процесів в їх динаміці.

Часто якість освіти поділяють на два взаємопов'язані компоненти - якість знань, і якість освітнього процесу. Ці два поняття знаходяться в причинно-наслідковому зв'язку і виступають як нерозривна пара - мета і засіб.

Важливою проблемою, яку вирішує система з візуального моніторингу якості отримання знань, є надання якомога точнішої оцінки та рекомендацій, про якість засвоєних знань, під час навчання онлайн.

Тому було запропоновано розробити веб-систему з якості оцінювання отриманих знань, яка за допомогою відеофіксації, та можливістю відправки відеоматреіалів на сервіс з розпізнавання емоцій та концентрації уваги, буде максимально точно давати зворотній відгук про якість засвоєння інформації. Основні завдання розробки системи:

— Надання доступу до перегляду курсів та уроків.

— Проходження уроків та тестів.

— Відеофіксація уроку.

— Обробка результатів тесту та відео.

— Можливість відправки відеофайлів на ресурс з розпізнавання емоцій.

— Надання результату та рекомендацій після уроку.

— Доповнення системи новими курсами та уроками.

— Моніторинг успішності студентів.

Висновки до розділу 1

Задача веб системи моніторингу якості отримання знань користувача полягає в наданні доступу до курсів з уроками, в яких користувач може отримувати нові знання, проведення тестування та подальший аналіз результатів тесту, та обробки відеозапису уроку. Водночас дана веб система надаватиме змогу викладачам додавати нові курси та уроки, змінювати або видаляти старі, а також переглядати результати користувачів, та надавати додаткові коментарі.

2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ МОНІТОРИНГУ ЯКОСТІ ОТРИМАННЯ ЗНАНЬ

Розглянемо існуючі веб системи з моніторингу якості отримання знань .

2.1 Проект в сфері масової онлайн-освіти – Coursera.

Coursera - всесвітня онлайн платформа для дистанційного навчання, заснована в 2012 році Стенфордськими професорами інформатики Ендрю Ng і Дафною Коллер.

Coursera працює з університетами та іншими організаціями, пропонуючи онлайн-курси, спеціалізації та ступені з різних предметів, таких як інженерія, наука про дані, машинне навчання, математика, бізнес, інформатика, цифровий маркетинг, гуманітарні науки, медицина, біологія, соціальні науки, та інші.

Coursera дає вам можливість отримати знання і досвід від кращих викладачів у своїй сфері діяльності. Більш того, багато хто з них крім викладання також мають власний бізнес.

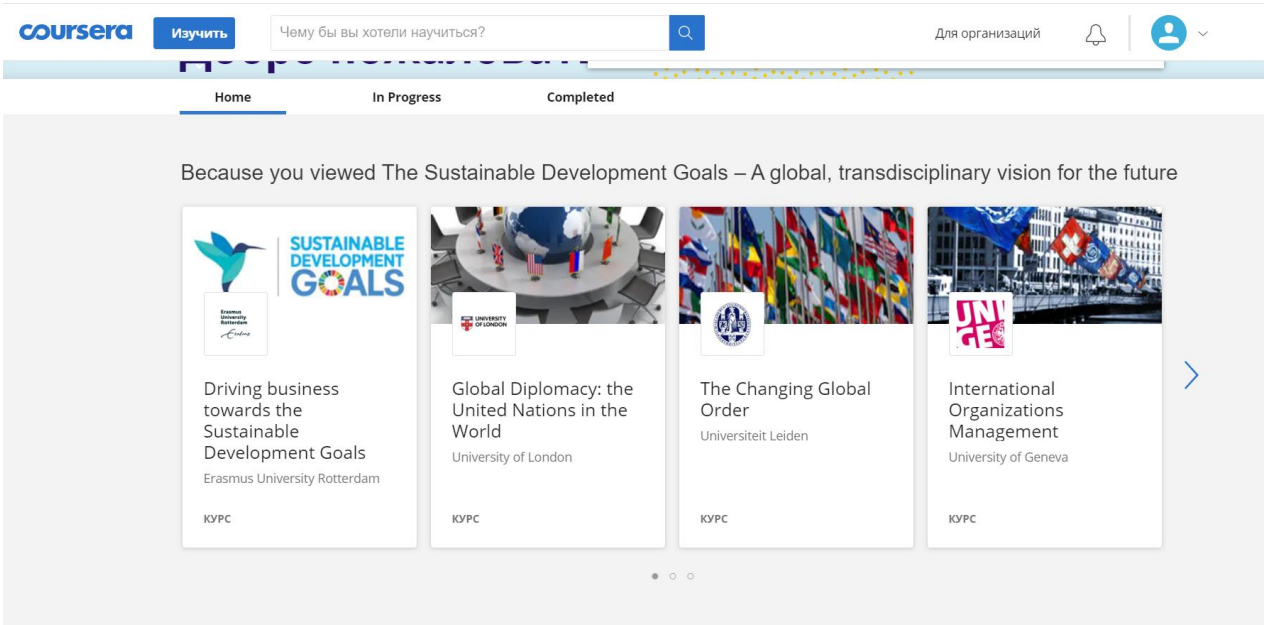


Рисунок 2.1 – Інтерфейс веб системи Coursera.

Проте обмеження цього формату очевидні. Срібної кулі не вийшло, і замінити одним приголомшливим викладачем сто звичайних не вийде. Coursera не підходить для отримання обсягу знань, порівнянного зі стандартним бакалавром - там все життя піде, поки все збереш, і вона не може замінити здобуття звичного диплому. А також один із мінусів – це те що дана веб система не надає повноцінного моніторингу отриманих знань.

2.2 Американська платформа онлайн-навчання - Udemy

Udemy, заснована в травні 2010 року, є американською платформою онлайн-навчання, призначеною для професійних дорослих і студентів. Станом на січень 2020 року на платформі навчаються понад 50 мільйонів студентів і 57 000 викладачів на більш ніж 65 мовах. Було зареєстровано більше 295 мільйонів студентів. Студенти і викладачі з більш ніж 190 країн, а 2/3 студентів знаходяться за межами США.

Студенти беруть курси в основному як засіб підвищення професійних навичок. Деякі курси привертають увагу до технічної сертифікації. «Удемо» доклав особливих зусиль для залучення корпоративних тренерів, які прагнуть створити курсову роботу для співробітників своєї компанії. Станом на 2020 рік на сайті є понад 150 000 курсів.

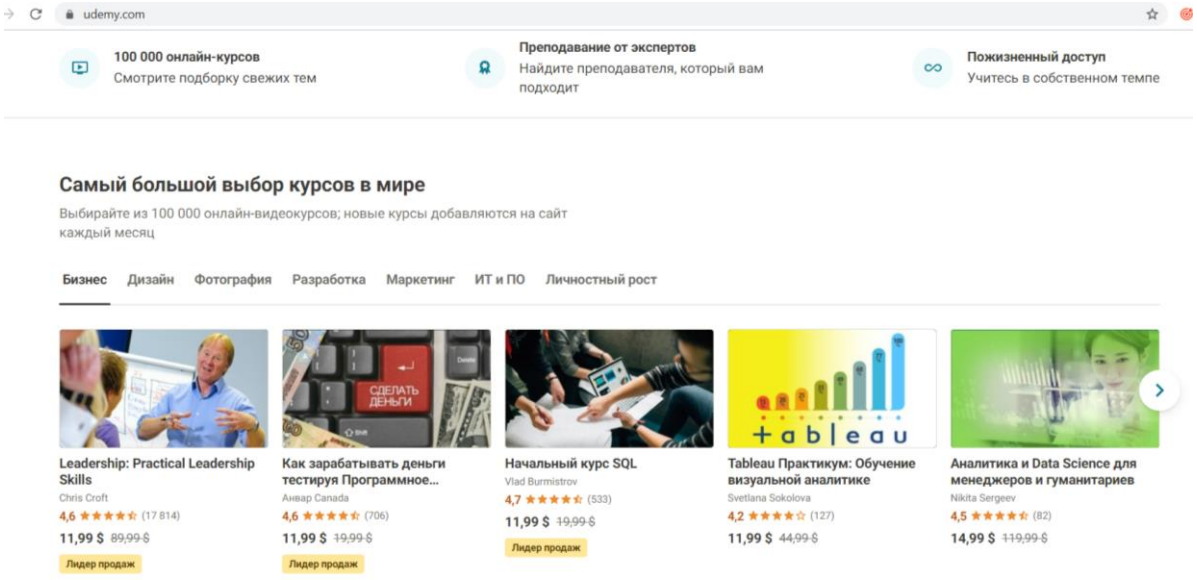


Рисунок 2.2 – Інтерфейс веб системи Udemy.

Серйозною проблемою є той фактор, що будь-який користувач може стати викладачем, тобто безумовно буде багато неякісних курсів. У більшості авторів курсів добрі наміри, але оскільки вони не завжди є експертами в цій галузі, то і навчати вони можуть неякісно.

Висновки до розділу 2

Таким чином, було проаналізовано дві найпопулярніші веб-системи з надання доступу до знань та моніторингу якості засвоєння цих знань.

В результаті аналізу можна виділити наступне:

— Веб платформа Coursera надає безкоштовний доступ до курсів, які написані іменитими викладачами. Але курсів надто мало і тільки деякі викладачі можуть публікувати там свої курси, а також у даної системи не реалізований моніторинг якості отриманих знань;

— Udemy популярна платформа онлайн-навчання, у якій зібрано десятки тисяч курсів. Але так як кожен користувач може створити свій курс, купивши його, можна дуже розчаруватись, оскільки не кожен зможе вміло та доступно викласти інформацію.

3 МОДЕЛЮВАННЯ СИСТЕМИ

Протягом кількох останніх десятків років виникли нові проблемаи розвитку освіти. Знання старіють кожні три-п'ять років, а технологічні знання – кожні два-три роки. Об’єм знань випускників вищих навчальних закладів збільшується вдвічі кожні три-чотири роки. Якщо освітні технології залишити на теперішньому рівні, то кваліфікація фахівців буде об'єктивно нижчою від критеріїв сучасного ринку праці. Сприйняття знань учнями за допомогою інформаційних і комунікаційних технологій за найнижчими оцінками на 40-60% швидше, ніж з використанням звичайних технологій.

Нововведення притаманні для будь-якої професійної діяльності людини і тому, безумовно, є предметом вивчення, розгляду і впровадження. Інновації не виникають самі собою, вони є продуктом наукових пошуків, новітньої педагогічної практики окремих викладачів і навіть цілих груп. Даний процес не може бути некерованим або стихійним, він потребує управління.

Серед видів нововведень в галузі освіти виділяється застосування технологій дистанційного навчання. Метою використання ДОТ (дистанційних освітніх технологій) є надання учням які навчаються можливості освоєння освітніх програм безпосередньо за місцем проживання учня або його тимчасового перебування.

Під час застосування онлайн-навчання в освітньому процесі роль педагога все більше матиме рекомендативне значення, разом з тим виділяючи викладачеві більше часу для наукової діяльності.

Ще однією беззаперечною прерогативою дистанційних технологій є ціна навчання. Експерти дослідили, що інтерактивно телекомунікаційне викладання виходить на 20-25% дешевше ніж традиційне. Компанія «Microsoft» підрахувала, що вартість онлайн-навчання може впасти як мінімум у два рази порівняно з традиційним навчанням, оскільки педагог має можливість давати уроки перебуваючи у будь-якому місці. При цьому немає необхідності в особливому комп'ютерному

обладнанні. Це вкрай важливо в екстрених ситуаціях, таких як недуга викладача, його відрядження або просто з будь-які інші причини, які унеможливлюють присутність на заняттях. Можна досягнути економії і за допомогою інших чинників. Адже під час онлайн навчання можна бути впевненим, що всі учні використовують однакові і крім того, найсвіжіші, навчально-методичні матеріали. Адже оновлювати навчальні посібники з допомогою Інтернету набагато легше.

Може здатися, що застосування СДН неефективне тому що без візуального контакту з викладачем не можна з точністю зробити висновок про підготовленость студента. Задля вирішення цієї проблеми, було придумано створити систему з візуального моніторингу якості отримання знань. Ця система буде слідкувати за активністю студентів, дозволяючи викладачам легко зробити висновок про здібності студента, а самим студентам надавати рекомендації що до покращення їхніх результатів.

Беззаперечний привілей індивідуального підходу під час дистанційного навчання показується у дослідженнях, проведених американськими розробниками методу «SCORM». Вчені зрівнювали індивідуальне навчання і заняття в класі. Було відзначено такі особливості:

- В середньому, на одного студента групи під час роботи в класі доводиться 0.1 питання на годину. При індивідуальному навчанні студент має можливість поставити або відповісти на 130 питань в годину;
- Для 97% студентів продуктивність при індивідуальній роботі підвищується на 55%.

Також важливо виділити, що під «індивідуальним навчанням» розуміється метод навчання, коли студент працює тільки з навчальною програмою. Індивідуальна робота, безсумнівно, дає кращий результат. Тому дистанційне навчання, завдячуючи технологічному прогресу, набуває все більшої ефективності і доступності. А з допомогою системи візуального контролю, онлайн навчання буде мати ще більше переваг над традиційним.

3.1 Загальні вимоги

Системи дистанційного навчання призначені для навчання за межами навчальних місць. Мета використання систем це підвищення якості навчання, створення короткострокових курсів для підвищення кваліфікації учнів, а також для накопичення та застосування знань в різних наукових галузях науки, що досліджуються в різних навчальних закладах.

Реалізація повинна являти собою програмний продукт, який дозволяє створити середовище для вивчення нової інформації шляхом проходження курсів та надання зворотного коментаря по якості отриманих знань, за результатами тестів та відеофіксації. Взаємодія з системою має виконуватись через глобальну мережу Інтернет із застосуванням протоколів прикладного рівня.

3.2 Користувачі системи

Користувачів системи варто розділити за обов'язками і надати їм певні права доступу до функцій системи. Було вирішено створити два види користувачів: викладача і студента.

Викладачі повинні мати доступ до функцій створення інтерактивних курсів, а також додавання та редагування в інформаційного матеріалу в системі. Повинна бути доступна можливість здійснення контролю успішності студентів.

Студенти матимуть можливість доступу до курсів, уроків, проходити тести, отримувати зворотні відгуки та поради, щодо результатів тесту, а також мати можливість консультуватися з викладачами.

3.3 Модуль накопичення даних

У системі потрібно створити механізми збереження навчальних матеріалів, можливості опису, редагування та пошуку. Усі наявні дані необхідно чітко

структурувати і представляти у вигляді курсу. Курс має містити уроки і закінчуватися тестуванням учнів.

Модуль накопичення інформації необхідно консолідувати і створити підтримку введення даних з інших баз в яких зберігаються дані, розповсюджених у всесвітній мережі Інтернет. Необхідно продумати майбутній процес модернізації даних на наступних етапах з використанням метамов.

Оскільки в системі будуть зберігатись особисті данні та відеозаписи студентів потрібно звернути увагу на безпеку. Паролі повинні бути зашифровані, а в назві відеозапису не повинна містить інформація про користувача.

3.4 Проведення освітнього процесу

Доступ до освітніх ресурсів, створених викладачами, повинен здійснюватися тільки після реєстрації користувача. Вивчивши лекційний матеріал, учні зобов'язані виконати тестові завдання. Тест має містити питання і варіанти відповідей до них. Після проходження тестів їх результати та відеофіксація користувача під час уроку зберігаються в системі.

Викладачеві, відповідно, надається доступ до результатів групи. Протягом усього курсу студенти повинні мати можливість консультуватися з викладачем. Підтримка редагування курсів, уроків системи обов'язкова.

3.5 Моделювання інтерфейсу користувача

Система дистанційного навчання повинна мати водночас дружній та інтуїтивно простий зовнішній вигляд інтерфейсу. Розробка візуального оформлення має здійснюватися з врахуванням можливостей усіх вікових груп користувачів. Користувач з першого погляду повинен зрозуміти, як користуватись системою.

Потрібно організувати модуль підтримки користувачів. Необхідно створити платформу для розробки інформації для довідок по системі і для технологій онлайн-навчання в цілому.

3.6 Можливості модернізації

В системі дистанційного навчання повинна бути реалізована можливість модернізації і заміни модулів. Модель системи повинна дозволяти адаптувати її до умов, що змінюються в процесі її експлуатації умовами і допускати поетапну модернізацію окремих компонентів.

Для досягнення цих цілей в процесі програмування необхідно використовувати об'єктно-орієнтований стиль і дотримуватися вимог до процесу програмування.

3.7 Відеофіксація

Головною проблемою дистанційного навчання є відсутність візуального контролю зі сторони викладача за якістю отримання знань учнів. Під час знаходження учня в домашній обстановці або в іншому місті, яке не є спеціалізованим під навчання та без нагляду викладача, у багатьох випадках, учні матимуть фактори, які будуть відволікати від навчання, тим самим погіршуючи якість отримання знань. Проаналізувавши цю проблему було вирішено створити систему саме з візуального моніторингу, тим самим приблизити дистанційне навчання до традиційного.

Відеофіксація буде проводитись за допомогою веб-камери користувача. Перед початком уроку користувача буде проінформовано про те, що система буде вести відеофіксацію під час уроку.

Після закінчення тестування, відеозапис з уроком, а також результати та час проведений за вивченням уроку, будуть зберігатись на сервері. З подальшою можливістю відправки відео на сервіс розпізнавання емоцій, який буде надавати інформацію про те на скільки уважно користувач вивчав урок.

Висновок до розділу 3

Підводячи підсумок, можна виділити основні вимоги до системи з візуального моніторингу якості отримання знань:

- ідеологія і методи онлайн навчання не повинні відрізнятися від традиційних;
- системний інтерфейс має бути інтуїтивно зрозумілий усім користувачам;
- у системі необхідно забезпечити стійкість і безпечність при роботі;
- система повинна мати зручність при використанні;
- необхідно передбачити фактор розширювання і масштабування системи.
-система має максимально точно надавати результати та поради користувачам, за рахунок обробки результатів тесту та відео матеріалу.

Отже, найголовнішою вимогою до системи з візуального моніторингу якості отримання знань є те, що процес дистанційного навчання повинен якомога менше відрізнятися від традиційного навчання.

4 ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Аналізуючи поставлену задачу та методи її вирішення, було обрано програмну архітектуру на основі веб-технологій. Основною прерогативою веб-застосунку над рештою варіантів є наявність універсальності та можливість виконання на різних гаджетах без прив’язки до операційної системи (браузер та його віртуальна машина приймає роль цільової універсальної операційної системи).

4.1 Вибір архітектури програмного комплексу

Для виконання поставленого завдання було прийнято рішення використовувати триланкову архітектуру. Її складовими є наявність таких компонентів: база даних, клієнт і сервер. Схема архітектури програмного комплексу зображена на рисунку 3.1.

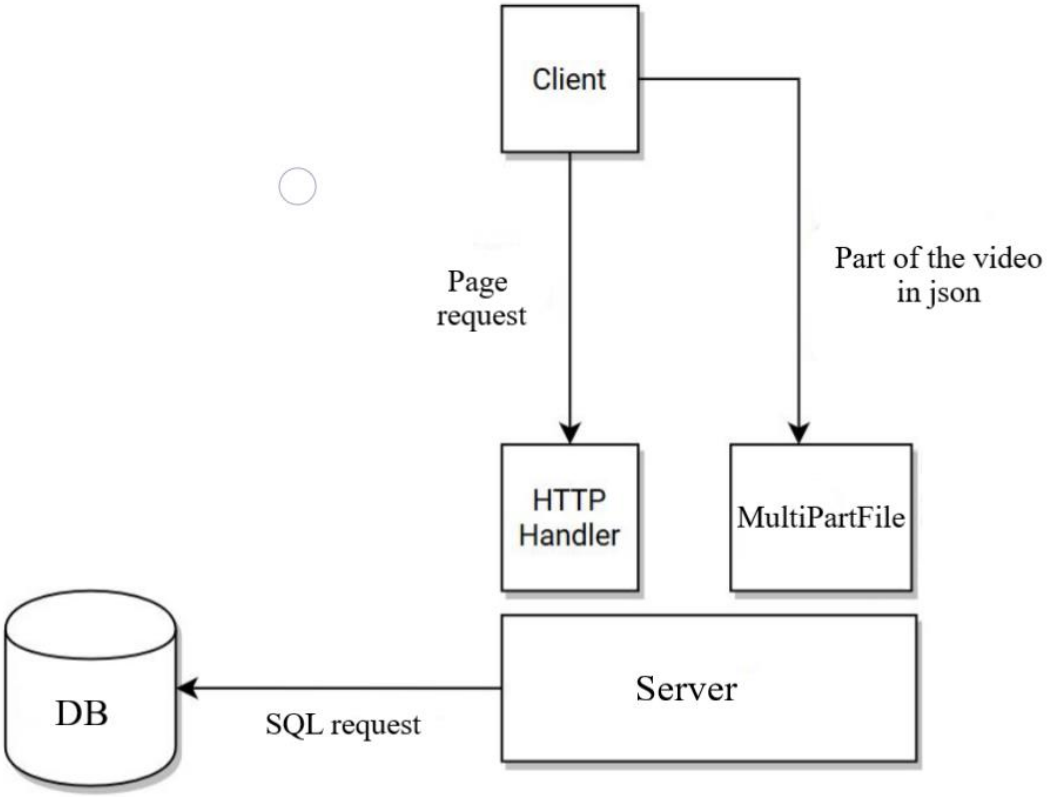


Рисунок 4.1 – Архітектура програмного комплексу

Основним центром програмної системи є сервер. Він зосереджує в собі основну бізнес-логіку та логіку зв’язку з базою даних. На сервері відбувається аутентифікація користувача задля надання йому індивідуального доступу до програмного продукту. Лише сервер зв’язує користувача та базу даних, для унеможливлення взлому даних та їх використання не по призначенню. Щоб користуватися системою, необхідно бути авторизованим, отже ця логіка знаходиться на рівні серверу, так як на рівні користувача існує ризик підміни прав доступу або інші способи неконтрольованого доступу до інформації користувачів.

При використанні програмного комплексу користувач здійснює взаємодію з клієнтським додатком, яким у даному випадку є веб-сайт. Інтерфейс користувача, за допомогою якого виконується безпосередньо налаштування програми, а також перегляд результатів роботи. Окрім цього на рівні користувача виконується попередня обробка інформації перед відправкою на сервер і для виконання опрацювання результатів від сервера. Ще на цьому рівні виконується перший етап ідентифікації користувача задля недопущення неконтрольованого доступу до програми.

Також під час проходження уроку, на рівні користувача буде вестись відео зйомка, яка по деякому інтервалу буде відправляти частини відеофайлу, закодовані в байт код, на сервер, де відео буде декодуватись і зберігатись. По завершенню уроку до результатів користувача буде додаватись посилання на відеофайл.

Головним завданням на рівні бази даних є безпосередньо збереження даних, які сервер отримує для майбутнього застосування. Крім цього гарантується цілісність інформації за допомогою як зовнішніх звя’зків так і ключів. Також на рівні бази даних є можливість реалізовувати певну частину бізнес-логіки, якій не потрібно використання зовнішніх джерел інформації крім власне бази даних і її таблиць.

4.2 Опис архітектури серверу

Шаблон проектування — це архітектура, рішення що описує метод вирішення завдань, які часто виникають при розробці програмних комплексів чи додатків.

Задля реалізації серверу було обрано фреймворк, який є шаблоном проектування MVC.

MVC розшифровується як модель - представлення - контролер (від англ. Model-view-controller). Це спосіб організації коду, який передбачає виділення блоків, що відповідають за вирішення різних завдань. Один блок відповідає за дані додатки, інший відповідає за зовнішній вигляд, а третій контролює роботу додатка. Компоненти MVC:

- **Представлення**

Це та частина програми, яка представляє відображення даних. Представлення створюються на основі даних, зібраних з даних моделі. Представлення запитує модель про надання інформації, яку потрібно відобразити користувачеві.

Представлення також представляє дані з чатів, діаграм і таблиць. Наприклад, будь-яке представлення клієнта буде включати всі компоненти користувальницького інтерфейсу, такі як текстові поля, списки, що розкриваються і т.д.

- **Контролер**

Це та частина програми, яка обробляє взаємодію з користувачем. Контролер інтерпретує введення миші і клавіатури користувача, інформуючи модель і вид яких змін необхідно внести.

Контролер відправляє команди моделі для поновлення її стану (наприклад, збереження певного документа). Контролер також відправляє команди зв'язаному представленню, щоб змінити відображення представлення.

- **Модель**

Компонент моделі зберігає дані і пов'язану з ними логіку. Він представляє дані, які передаються між компонентами контролера або будь-які інший пов'язаний бізнес-логікою. Наприклад, об'єкт моделі буде витягувати інформацію про клієнта з бази

даних. Він маніпулює даними і відправляє їх назад в базу даних або використовує їх для візуалізації тих же даних.

Він відповідає на запит від представлення, а також відповідає на інструкції від контролера, щоб оновити себе. Це також найнижчий рівень шаблона, який відповідає за ведення даних.

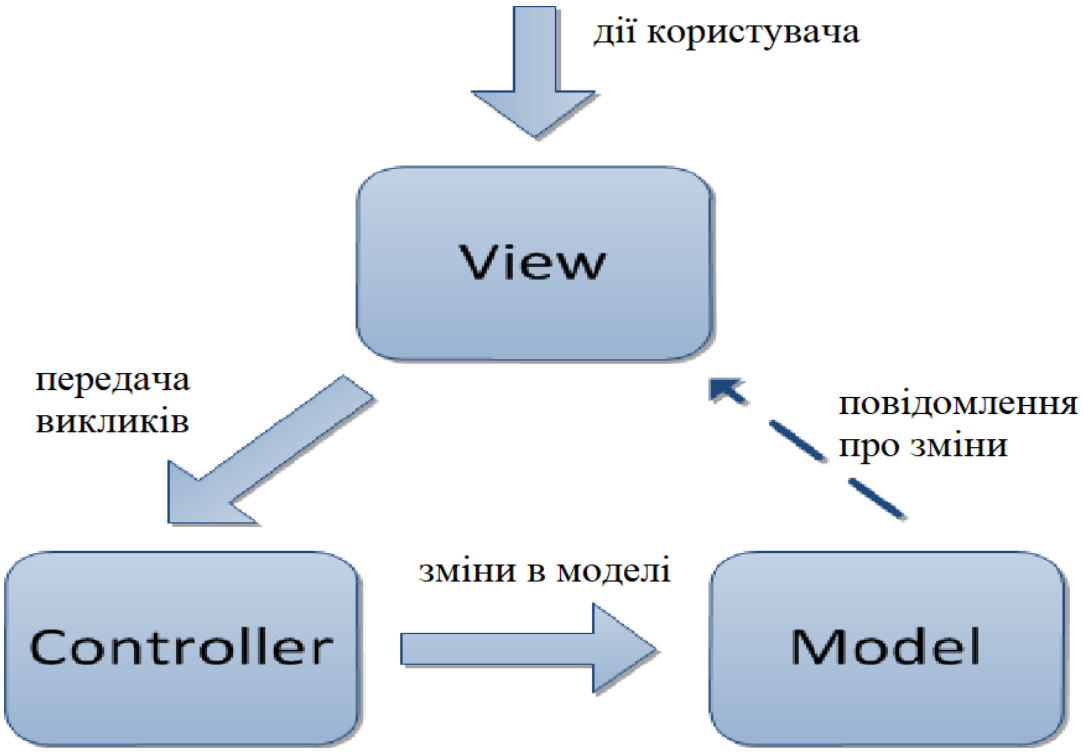


Рисунок 4.2 – Схема роботи MVC шаблону

Головною метою використання даної концепції є відділення бізнес-логіки моделі від представлення. Отже завдяки такому поділу зростає можливість багаторазового використання та універсальності системи. Окрім цього для великомасштабних проектів використовується шаблон MVC чи його похідні які дозволяють забезпечити тестування різних компонентів програмного комплексу.

4.3 Опис архітектури клієнтського застосунку

Для реалізації клієнтського застосунку було використано механізм шаблонів Apache FreeMarker.

FreeMarker - це механізм шаблонів: бібліотека Java для генерації текстового виведення веб-сторінки HTML. На основі шаблонів і зміни даних. Шаблони написані на мові шаблонів FreeMarker (FTL), яка є простою спеціалізованою мовою. Зазвичай для підготовки даних використовується мова програмування загального призначення. Потім Apache FreeMarker відображає підготовлені дані з використанням шаблонів рис.3.3. У шаблоні ви зосереджуєтесь на тому, як представляти дані, а по-за шаблоном ви зосереджуєтесь на тому, які дані представляти.

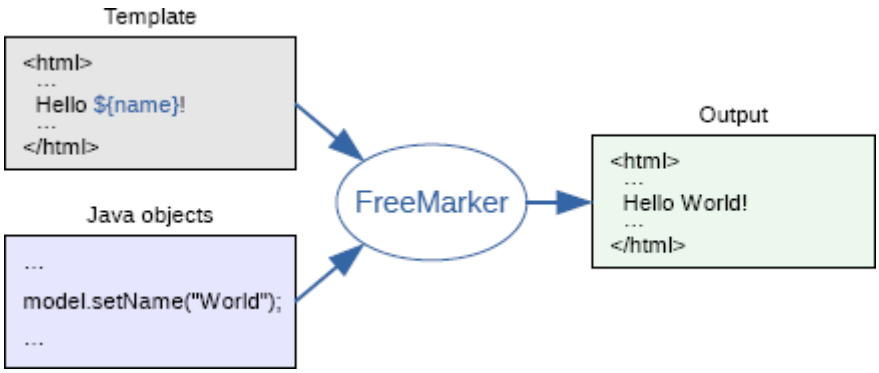


Рис 4.3 Робота яку виконує FreeMarker

Кілька основних моментів FreeMarker:

- Потужна мова шаблонів: умовні блоки, ітерації, призначення, строкові та арифметичні операції і форматування, макроси і функції, включаючи інші шаблони, екранування за замовчуванням (необов'язково) і багато іншого.
- Багатоцільовий і легкий: нульові залежності, будь-який формат виведення, можна завантажувати шаблони з будь-якого місця (підключається), безліч параметрів конфігурації
- Інтернаціоналізація / з урахуванням локалізації: Чутливий до локалі номер і форматування дати / часу, локалізовані варіанти шаблонів.

- Універсальна модель даних: об'єкти надаються шаблоном у вигляді дерева змінних через адаптери що підключаються, які вирішують, як шаблон їх бачить.

4.4 Опис інструментів розробки

Програмна система побудована по принципах триланкової архітектури програми. Це означає що кожен із рівнів даної архітектури спроектовано з використанням різноманітних технологій і основною метою є створення мультиплатформного рішення із застосуванням відкритих технологій.

Для створення клієнтського рівня було застосовано таку сукупність технологій: мова програмування JavaScript, механізм шаблонів FreeMarker (описаний в пункті 3.3), мова розмітки HTML та формальна мова опису зовнішнього вигляду документа CSS.

Сучасний JavaScript - це «безпечна» мова програмування. Він не надає низькорівневий доступ до пам'яті або процесору, тому що спочатку був створений для браузерів, які не потребують цього. «JavaScript» у переважній більшості випадків працює як вбудована мова для програмного доступу до об'єктів додатків.

Наприклад, в браузері JavaScript може:

- Додавати новий HTML-код на сторінку, змінювати існуючий вміст, модифікувати стилі.
- Реагувати на дії користувача, клацання миші, перемістити вказівник, натискати клавіші.
- Відправляти мережеві запити на віддалені сервера, завантажувати і завантажувати файли (технології AJAX і COMET).
- Отримувати і встановлювати куки, задавати питання відвідувачеві, показувати повідомлення.
- Запам'ятовувати дані на стороні клієнта («local storage»).

Найширше використовується у браузерах в якості мови сценаріїв з метою збільшення інтерактивності веб-сторінок. У статті «Найбільш незрозуміла мова

програмування в світі стала найпопулярнішою в світі мовою програмування »Дуглас Крокфорд стверджує, що лідируючу позицію «JavaScript» зайняв в зв'язку з розвитком «AJAX», оскільки браузер став переважаючою системою доставки додатків.

Веб-додатки - це додатки на основі веб-технологій, у яких функціональне наповнення забезпечуються сервером та доставляються через мережу Інтернет. Традиційна модель веб-додатку працює таким способом: клієнтською програмою відправляється HTTP-запит на сервер; на сервері проводиться деяка обробка, після цього видається HTML-сторінка клієнту (рис. 4.4).

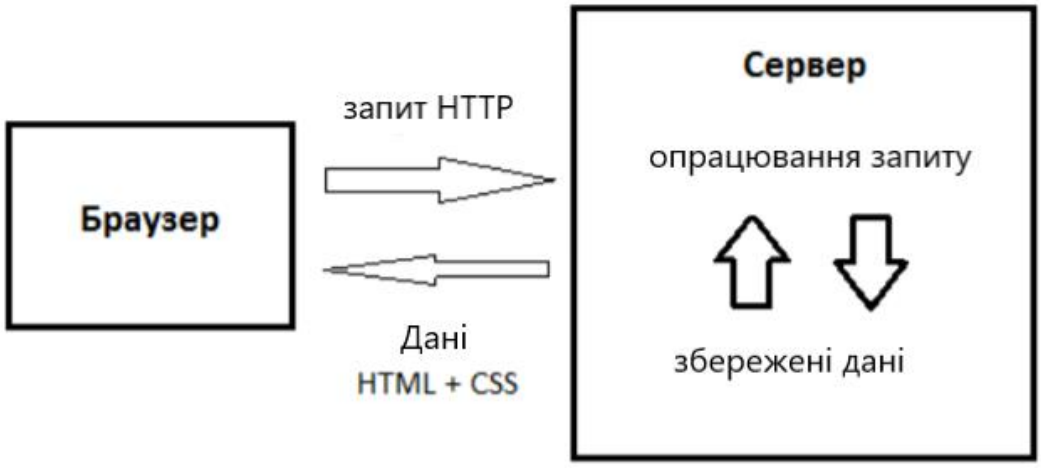


Рис 4.4

Істотним недоліком такого алгоритму є те, що клієнту доводиться чекати завантаження кожної наступної сторінки. Ідея AJAX, це скорочення очікування або зробити його зовсім непомітними. У розшифровці абревіатури AJAX і криється сенс технології. Asynchronous JavaScript and XML - технологія, що дозволяє здійснити передачу HTTP запиту без перезавантаження сторінки.

Для захвату відео на стороні користувача було використано метод getUserMedia Виклик getUserMedia () приймає MediaStreamConstraints як аргумент, який визначає параметри і / або вимоги до пристроїв захоплення і захоплених медіапотоків, таких як гучність мікрофона, дозвіл відео, яка камера включена (мається на увазі передня

або задня). Чи підтримується ваш браузер? getUserMedia () був доступний з Chrome 21, Opera 18, and Firefox 17, і зараз в Edge.

Під час зйомки відео, воно зберігається в байт код і через певний інтервал відправляється на сервер за допомогою протоколу HTTP, очищаючи після цього буфер, для запису наступного фрагменту. Потрібно правильно підібрати інтервал запитів, щоб не перевантажувати сервер запитами.

HTML - скорочення від "HyperText Mark-up Language" - перекладається як "Мова розмітки гіпертексту" (Гіпертекст - це текст, що не послідовно зв'язаний з іншими документами, тобто у вас є змога з першої сторінки документу перейти на останню). Іншими словами HTML - це мова розмітки, або ще один спосіб зберігання інформації. За допомогою HTML ти позначаєш текст, вказуючи своєму веб-переглядачу, як він має розуміти позначений текст.

CSS (Cascading Style Sheets) - це спеціальна мова (мова стилів), за допомогою якої описують вигляд документів (як і де відображати елементи веб-сторінки), написаних мовами розмітки даних. Найчастіше CSS використовується для документів, котрі розмічені мовою HTML, XHTML та XML.

На рівні серверу було вирішено застосувати такі технології: мова програмування Java, а саме фреймворк Spring, ORM для доступу до бази даних Hibernate і Spring JPA.

Java – є об'єктно-орієнтованою мовою. Вона підтримує поліморфізм, успадкування, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. Java є мовою з Сі-подібним синтаксисом і близький в цьому відношенні до C / C ++ і C #. Тому, якщо ви знайомі з однією з цих мов, то опанувати Java буде легше. Ще однією ключовою особливістю Java є те, що вона підтримує автоматичну збірку сміття. А це означає, що вам не треба звільняти вручну пам'ять від об'єктів що воикористовувалися раніше, як в C ++, так як збирач сміття це зробить автоматично за вас.

Spring Framework - забезпечує комплексну модель розробки і конфігурації для сучасних бізнес-додатків на Java - на будь-яких платформах. Ключовий елемент

Spring - підтримка інфраструктури на рівні програми: основна увага приділяється "водопроводу" бізнес-додатків, тому розробники можуть зосередитися на бізнес-логіці без зайвих налаштувань в залежності від середовища виконання.

Можливості:

- Впровадження залежності
- Аспектно-орієнтоване програмування, включаючи декларативне управління транзакціями
- Створення Spring MVC web-додатків і RESTful web-сервісів
- Та багато іншого...

Spring Data - додатковий зручний механізм для взаємодії з сутностями бази даних, організації їх в репозиторії, вилучення даних, зміна, в яких випадках для цього буде достатньо оголосити інтерфейс і метод в ньому, без імплементації.

Hibernate - бібліотека для мови програмування Java, призначена для вирішення завдань об'єктно-реляційного відображення (object-relational mapping - ORM). Дана бібліотека надає легкий у використанні каркас (фреймворк) для відображення об'єктно-орієнтованої моделі даних в традиційні реляційні бази даних. Метою Hibernate є звільнення розробника від значного обсягу порівняно низькорівневого програмування щодо забезпечення зберігання об'єктів в реляційної базі даних.

MultiPartFile – технологія за допомогою якої сервер отримує json файли від клієнта у двійковому коді, та автоматично конвертує його в файл, з подальшою можливістю збереження.

Наймасовішою системою керування базами даних є «MySQL», яка була розроблена компанією «Sun Microsystems» і наразі є у власності світового лідера у даній області - компанії «Oracle Corporation». Переважно «MySQL» застосовується як сервер, до якого підключаються віддалені або локальні клієнти. MySQL — вільна система керування реляційними базами даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена, як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем

керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. Важливою рисою баз «MySQL» є можливість реплікації даних, тобто створення резервних копій.

4.5 Обґрунтування вибору програмної реалізації

Під проектування системи було проаналізовано, а також і вивчено предметну область. Детально проаналізувавши було прийнято рішення розробляти систему, що заснована на веб-технологіях задля використання веб-браузером.

Застосовані на сервері технології було обрано за зручністю у їх використанні, відкритому вихідному кодові, популярності в наш час та з можливістю використання на пристрої з будь-якою операційною системою.

Фреймворк Spring надає змогу створювати програми мовою Java, на будь-якій операційній системі. Технологія Spring Boot MVC має надзвичайно великий обсяг функціональності для побудови веб-серверів різних рівнів складності, надаючи при цьому швидкодію та надійність. Також присутнє безвідмовне розгортання серверів з майбутнім масштабуванням без необхідності створення для цього особливої архітектури програмного комплексу. Щоб отримати доступ до бази даних було вирішено використати технологію Spring JPA та фреймворк Hibernate завдяки його зручності та багатогранності. Завдяки даному фреймворку є можливість відійти від реалізації конкретної бази даних та для створення відповідних моделей бази даних. Таких підхід дає можливість повністю сконцентруватися на побудові якісної бізнес-логіки.

На клієнтському рівні було обрано технології, які задовольняють такі самі вимоги як і серверні, але з тою різницею, що виконуватись будуть у браузері. Мова JavaScript була обрана через те, що вона зручна, дозволяє виконувати різні маніпуляції з даними та за допомогою технології ajax відправляти GET, POST запити на сервер. Для стилізації клієнтського додатку було обрано використати мови

розмітки та стилізації HTML, CSS відповідно. А також використаний механізм шаблонів FreeMarker, для генерації текстового виведення веб-сторінки HTML.

Базою даних було обрано MySQL по тій причині, що даний проект має відкритий вихідний код і має надзвичайно велику підтримку від розробників. Крім цього особливою перевагою є наявність високої швидкості роботи, надійність та велика кількість вбудованих функцій. Отже є можливість розгорнути цю базу даних на усіх сучасних операційних системах.

Отже обрані технології в їх поєднанні надають можливість побудувати якісний та надійний продукт, який є захищеним від патентних позовів зі сторони розробників, оскільки ці всі технології є покриті ліцензіями, які виключають цю можливість і безпосередньо надають доступ вихідних кодів своїх продуктів.

Висновки до розділу 4

Таким чином, в результаті аналізу предметної області, а також вимог, висунутих для проектування, було вибрано та обґрунтовано засоби реалізації системи з візуального моніторингу якості отримання знань. Визначено набір задач і методів їх рішення для побудови системи, а також проведено ряд експериментів для максимально стабільної та швидкої роботи системи.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Система моніторингу якості отримання знань буде складатися з трьох модулів, кожен з яких буде розбиватися на різну кількість функціональних підблоків. Кожен модуль дуже важливий і тільки разом данні модулі, можуть показати хороший результат і принести користь.

5.1 Опис функціональності системи

Веб система моніторингу якості отриманих знань містить у собі двох акторів – користувача(учня) системи та адміністратора(викладача);

На рисунку 5.1 представлена діаграма прецедентів.

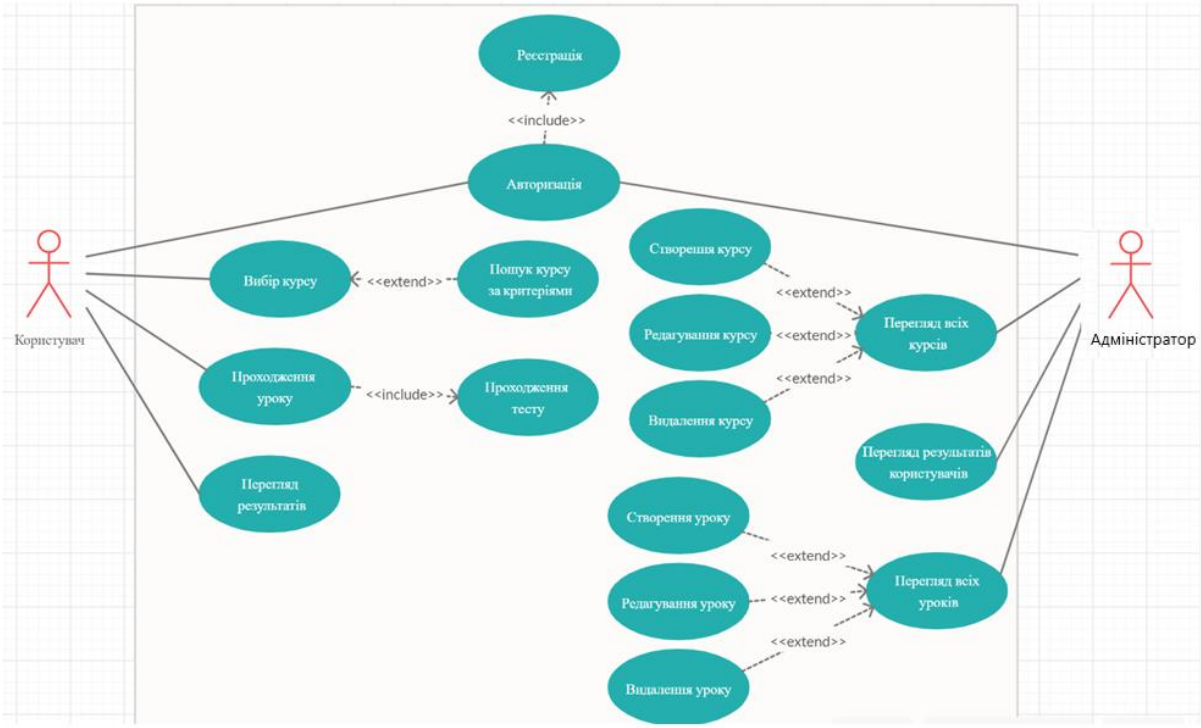


Рисунок 5.1 — Діаграма прецедентів

5.2 Концептуальна модель бази даних

База даних системи складається з 9 пов’язаних між собою таблиць бази даних, що утворюють об’єднаний інформаційний простір для збереження даних та отримання доступу до них.

Головними таблицями є:

- “Користувач” – міститься вся інформація про користувача.
- “Курс” – містить інформацію про курс, назву, кількість уроків та інше.
- “Урок” – зберігається інформація про кожен урок та тест до нього.
- “Тест” - зберігається інформація про тест, така як питання, відповіді, та номер уроку для якого розрахований тест.
- “Результат” – містить в собі дані про успішність, відеофікцсаію, затрачений час користувача.

Так існують другорядні таблиці такі як:

- “Зображення” – мітить посилання на зображення до уроку.
- “Посилання” – в даній таблиці зберігають посилання на відео до уроку, які знаходяться на відеохостнгу youtube.
- “Теги” – зберігаються теги до курсів.
- “Теми” – зберігаються теми всіх курсів.

Концептуальна модель бази даних приведена на рисунку 4.4.

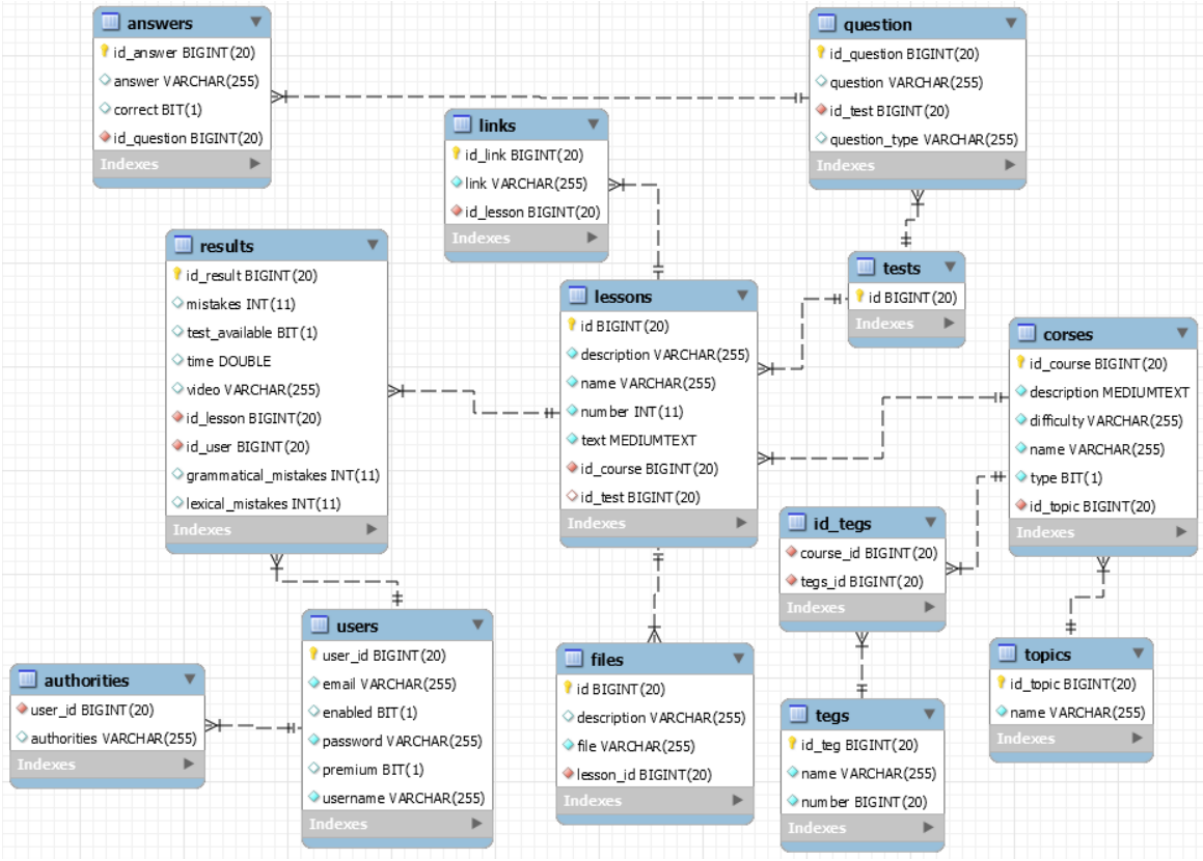


Рисунок 5.3 — Концептуальна модель БД

5.3 Опис таблиць бази даних

Для забезпечення доступу до інформації із бази даних для кожної із таблиць потрібно створити entity клас на мові програмування Java з властивостям, які будуть відображати поля таблиці в базі даних.

Repository надає інтерфейс що б отримати доступ і маніпулювати даними, які є збереженими в базі даних.

Клас Entity є аналогом таблиці в базі даних, поле моделі являється відповідною копією значення окремого стовпця рядка.

Окрім цього при застосуванні об'єктно-реляційної проекції надається можливість зручного отримання даних із таблиць, які мають зв'язки “багато-до-багатьох” і “один-до-багатьох”.

База даних системи реалізована за допомогою ORM Hiberante. Розглянемо більш детально структури основних таблиць бази даних.

Більш розширена інформація про структуру таблиць (ім’я, розмір і тип поля, описання поля) надана у таблицях 5.4 — 5.8.

Таблиця 5.4. Таблиця “Користувач”

Ім’я поля	**Тип**	**Опис поля**
Id	bigint(20)	Первинний ключ
UserName	varchar(100)	Логін користувача
Email	varchar(100)	Адреса електронної пошти
PasswordHash	varchar(10000)	Хеш паролю
Premium	boollean()	Доступ до преміум курсів

Таблиця 5.5. Структура таблиці “Курс”

Ім’я поля	**Тип**	**Опис поля**
Id	bigint(20)	Первинний ключ
Name	varchar(100)	Назва курсу
Description	varchar(500)	Опис курсу
Difficulty	varchar(10)	Складність
Type	bollean()	Тип(безкоштовний або преміум)

Таблиця 5.6. Структура таблиці “Урок”

Ім’я поля	**Тип**	**Опис поля**
Id	bigint(20)	Первинний ключ
Name	nvarchar(100)	Назва уроку
Description	nvarchar(450)	Короткий опис
Text	mediumtext	Текст уроку
TestId	bigint(20)	Ключ тесту

Таблиця 5.7. Структура таблиці “Тест”

Ім’я поля	**Тип**	**Опис поля**
Id	bigint(20)	Первинний ключ
Name	nvarchar(100)	Назва тесту
Question	nvarchar(500)	Питання
Answer	nvarchar(200)	Відровідь

Таблиця 5.8. Структура таблиці “Результат”

Ім’я поля	**Тип**	**Опис поля**
Id	bigint(20)	Первинний ключ
Points	int()	Бали
Time	data()	Витрачений час
Video	nvarchar(100)	Посилання на відео
Result	nvarchar(500)	Результат
UserId	bigint(20)	Первинний ключ еористувача

5.4 Розробка кабінету користувача

Кабінет користувача – є головним робочим місцем користувача у системі. Він містить у собі інші підмодулі, що виконують головні функції системи.

До даних модулів належать:

— модуль перегляду всіх доступних курсів;

— модуль проходження курсів/уроків;

— модуль тестів;

— модуль перегляду результатів

5.4.1 Модуль кабінету викладача

Даний модуль дозволяє адміністратору:

— створювати, переглядати, редагувати курси та уроки;

— створювати тести;

— переглядати результати користувачів;

5.4.2 Модуль обробки результатів

Даний модуль відповідає за відправку відеофіксації на систему відслідковування емоцій та отримання результатів:

— відеозйомка користувача під час проходження уроку та тесту;

— відправка даних на систему відслідковування емоцій;

— отримання та структуризація результату.

5.5 Комунікація з клієнтською частиною

Для комунікації серверу і клієнту використовується протокол HTTP. Протокол HTTP слугує для обробки запитів сторінок додатку та ініціації сесії. Приклад сесії зображено на рисунку 4.7.

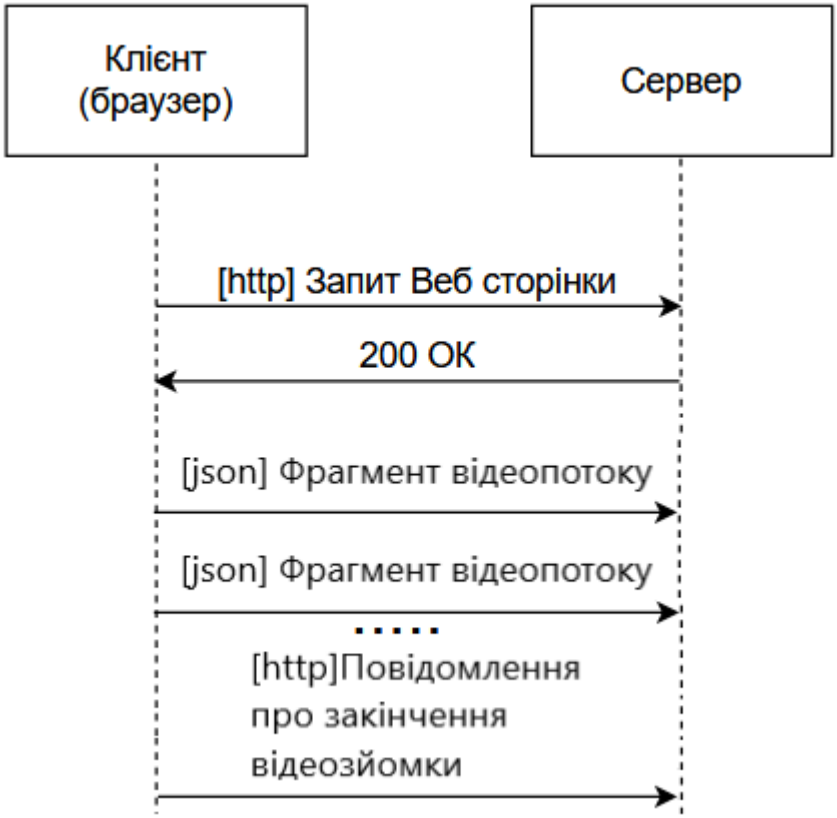


Рисунок 5.9 – Комунікація за допомогою HTTP.

Клієнт відправляє частини відеофайлу на сервер конвертуючи їх в json. Сервер в свою чергу зберігає частини відео в один файл, а після повідомлення про закінчення відеофіксації зберігає ім’я відеофайлу до результатів користувача.

Висновки до розділу 5

У результаті проектування та розробки системи з візуального моніторингу якості отримання знань було створено додаток з клієнт-серверною архітектурою. Додаток надає можливість користувачам проходити курси та тестування, роблячи відео фіксацію, з подальшим отриманням результатів та рекомендацій. Викладачі водночас мають можливість створювати нові курси, та при бажанні редагувати попередні, а також переглядати результати користувачів. Користувач взаємодіє з додатком через веб-інтерфейс у браузері.

6 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Система з візуального моніторингу отримання знань розроблена у вигляді клієнт-серверного веб-додатку. Для коректної роботи додаток рекомендується розгорнути на віддаленому сервері та надавати доступ за допомогою посилання. А також, додаток можна розгорнути і локально на комп'ютері користувача. Для цього потрібно встановити JAVA SDK версії 8 або вище і середовище розробки, наприклад Eclipse або Intellij IDEA(використана у даному випадку). Потрібно відкрити проект у середовищі розробки та запустити на виконання (рисунок 6.1).

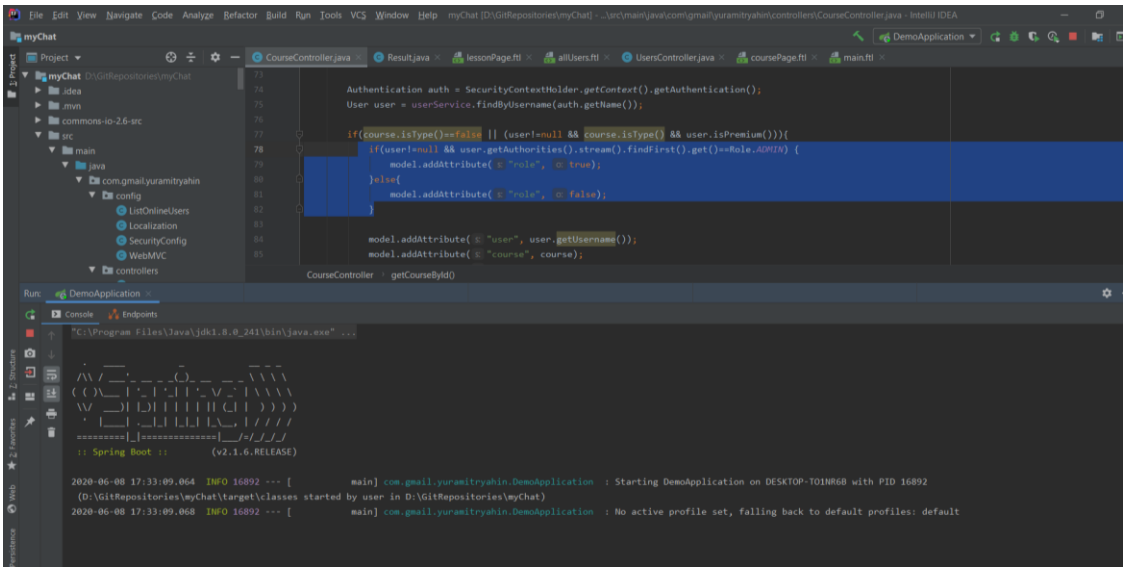


Рисунок 6.1 – Запуск сервера

Фреймворк Spring виконує автоматичний збір всіх класів і запускає систему. Після запуску серверу користувач може доступитися до клієнтської частини в браузері за адресою "http://localhost:8080/".

6.1 Інтерфейс користувача

Після відкриття додатку у браузері користувача зустрічає вікно авторизації(рис 6.2), якщо користувач ще не зареєстрований, він може перейти в меню реєстрації, інтерфейс якого зображений на рисунку 6.3.

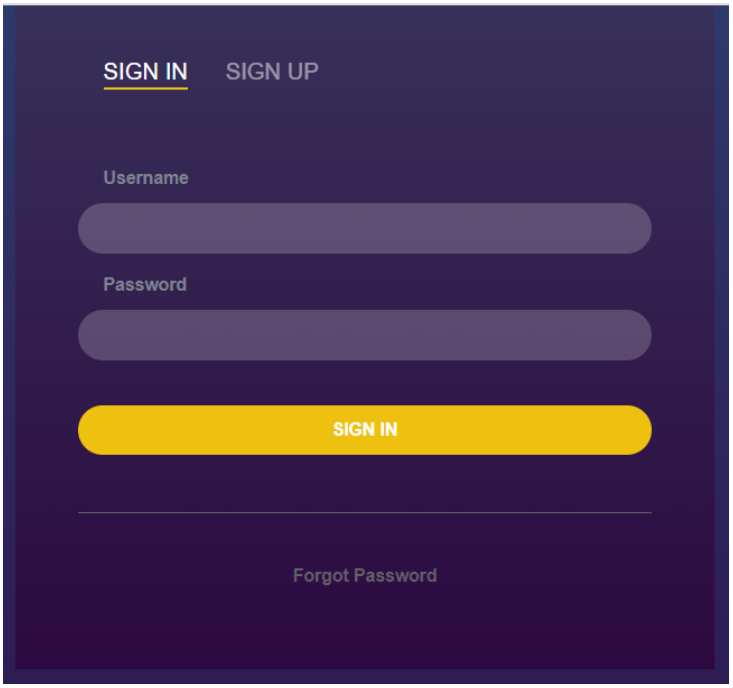


Рисунок 6.2 – Інтерфейс авторизації

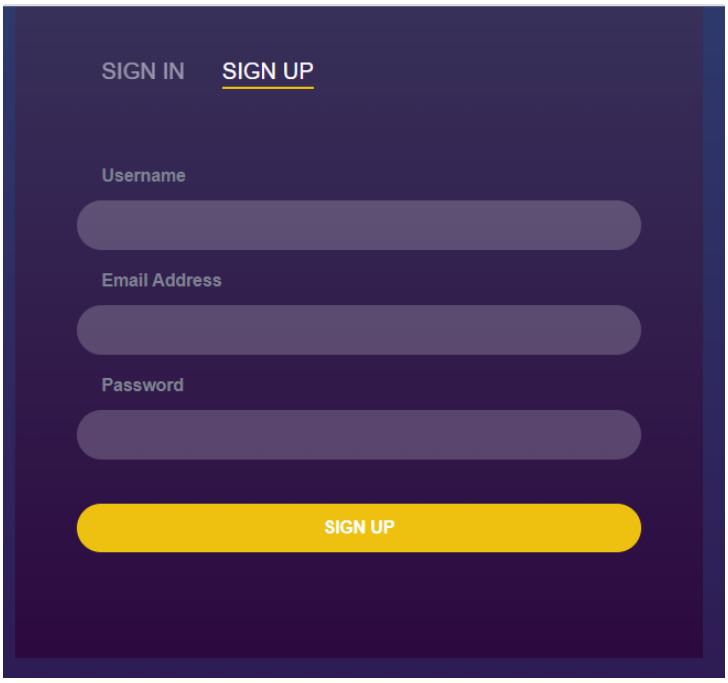


Рисунок 6.3 – Інтерфейс меню реєстрації

Після проходження авторизації користувача автоматично перекидає на головну сторінку, на якій розміщені курси. Для швидкого пошуку потрібного курсу у лівій частині головної сторінки розміщене меню пошуку за різними критеріями. Інтерфейс головної сторінки зображений на рисунку 6.4.

Рисунок 6.4 – Інтерфейс головної сторінки.

У системі існують два типи курсів, це безкоштовні і перміумні. Що б отримати доступ до преміумних курсів, користувачу потрібно звернутись до вчителя. Після того як користувач обере потрібний курс, відкривши його, йому будуть доступні всі уроки(рисунок 6.5).

Рисунок 6.5 – Сторінка курсу

Після натискання на потрібний урок, користувача буде перекинуто на сторінку з уроком, перед стартом заняття, користувача буде попереджено про початок відео зйомки(рисунок 6.6)

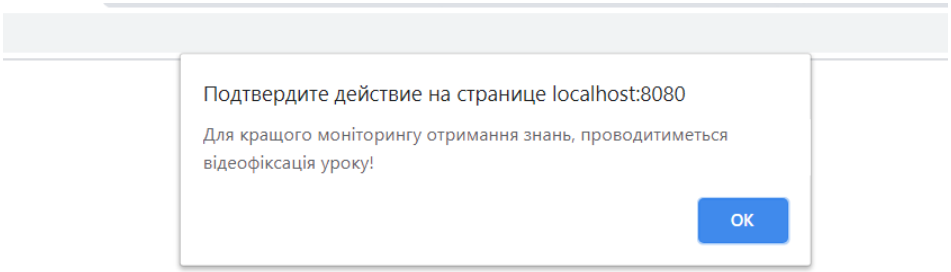
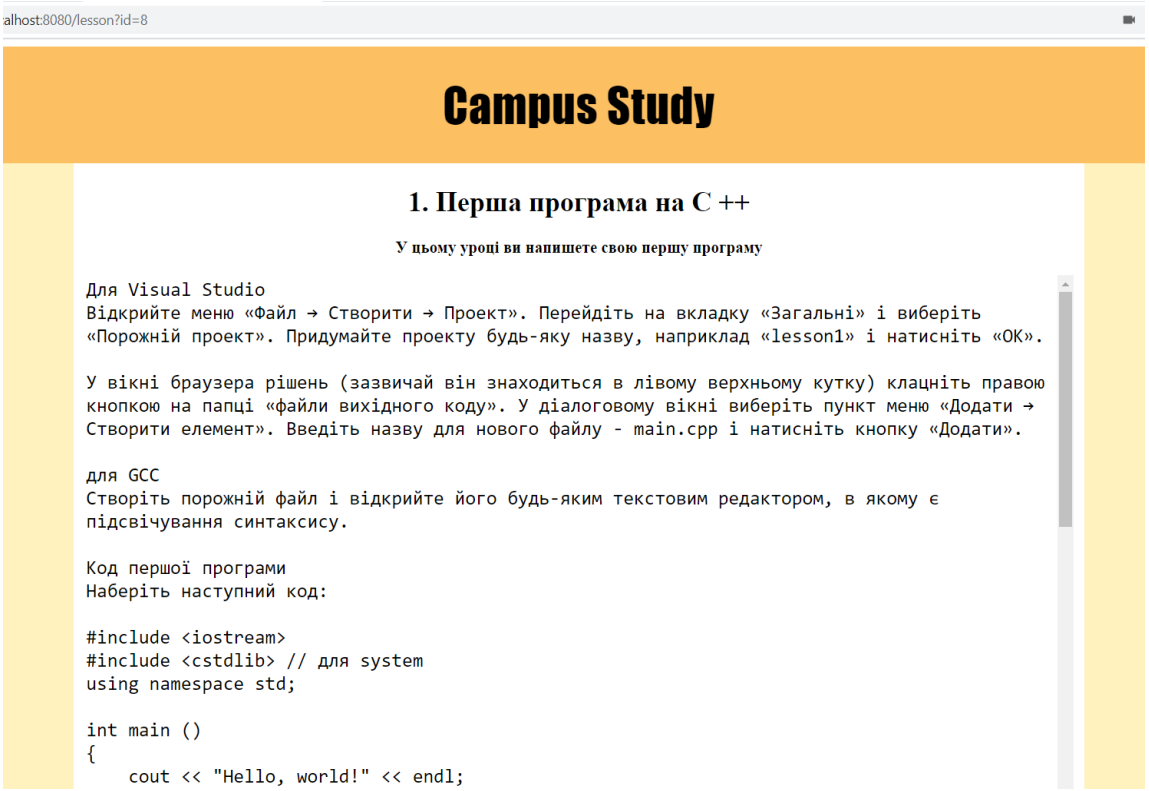


Рисунок 6.6 – Попередження про початок відео фіксації

Структура уроку складається із текстової інформації, картинок та, по можливості, з відео. Після опрацювання матеріалу уроку, користувач може завершити урок або пройти тест. Під час проходження тесту у користувача блокується інформація уроку. Інтерфейс уроку та тесту зображено на рисунках 6.7 і 6.8 відповідно.



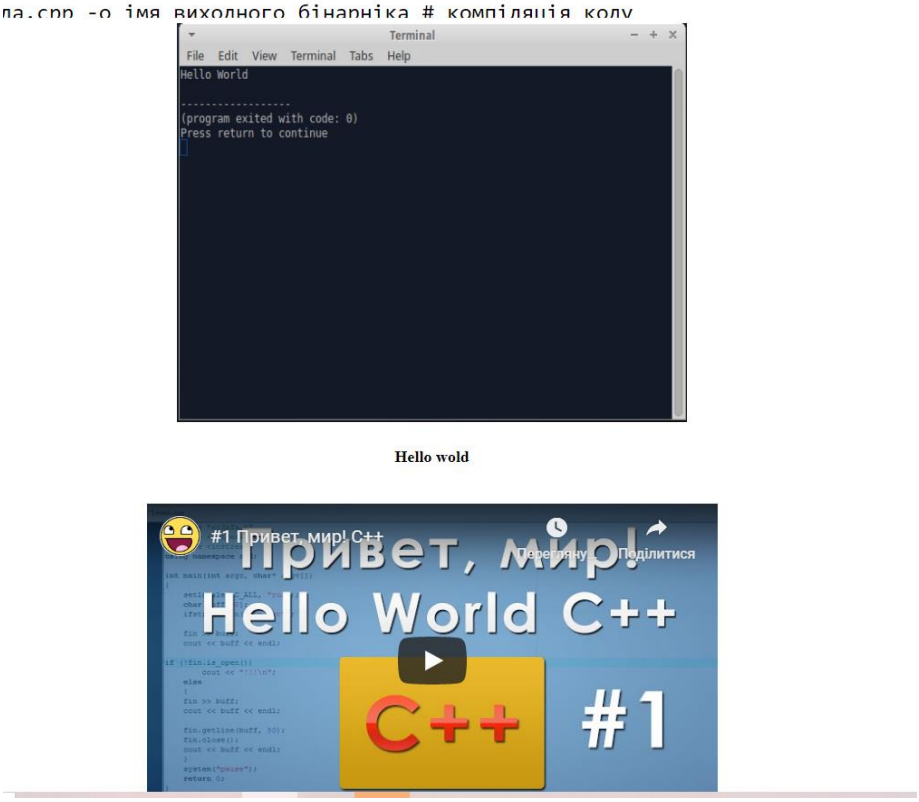


Рисунок 6.7 – Інтерфейс уроку

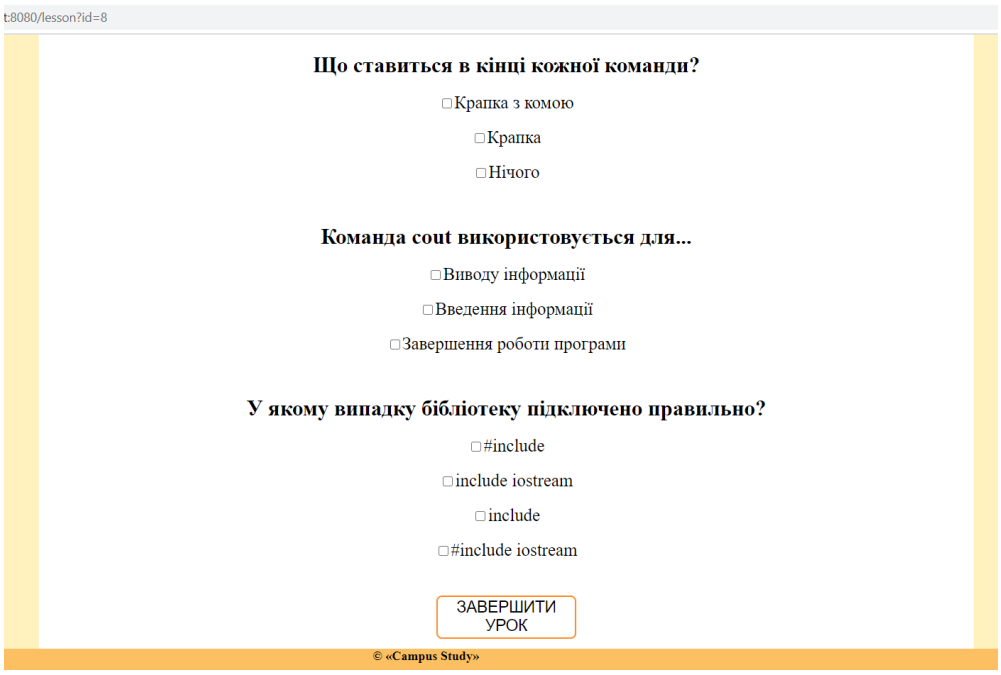


Рисунок 6.8 – Інтерфейс тесту уроку

Після натиснення кнопки завершення уроку, користувачу надається сторінка з результатами тесту у якому зазначено час проведений за уроком, кількість помилок та

рекомендації про те, як покращити результат в наступного разу. Приклад такого вікна зображено на рисунку 6.9.

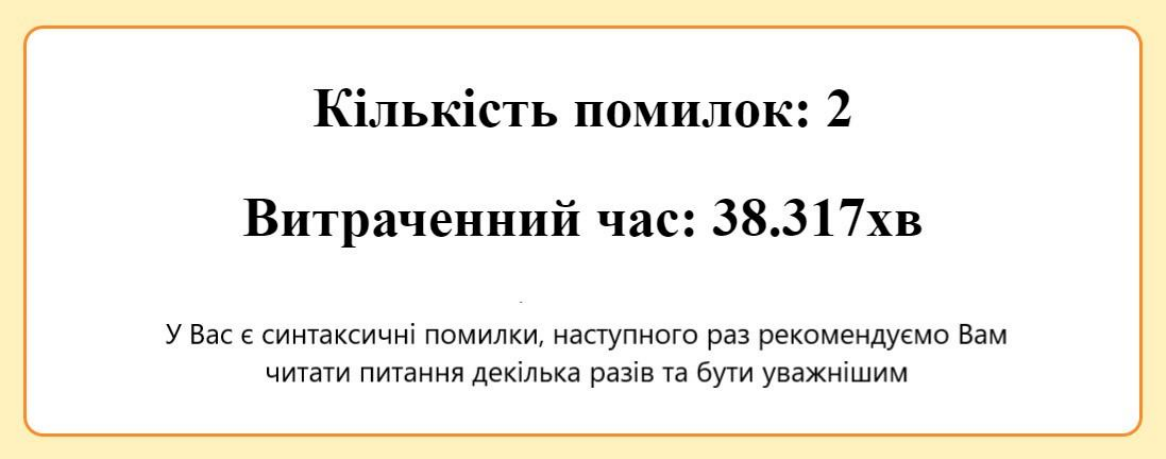


Рисунок 6.9 – Вікно результату уроку.

Також у користувача є можливість переглянути всі попередні результати.

6.2 Інтерфейс викладача

Після авторизації у системі у викладача крім функціоналу користувачів є також допоміжні можливості, на сторінці курсу у викладача з’являються кнопки редагування, видалення курсу та уроку(рис. 6.10).

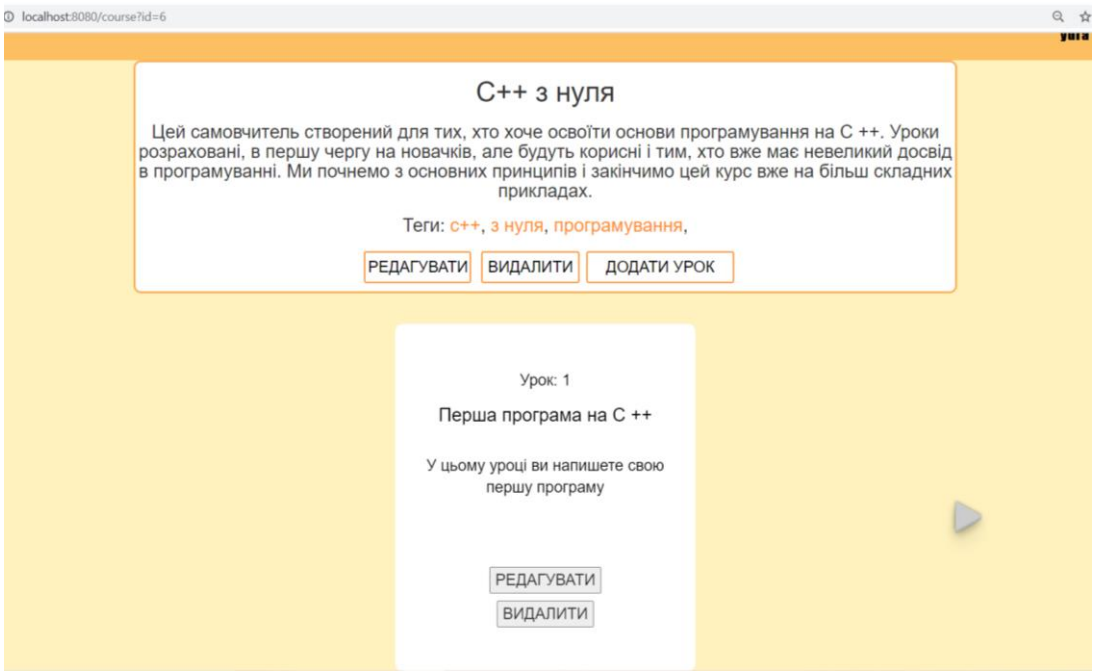


Рисунок 6.10 – Додаткові можливості викладача

Також є можливість додати курс, урок, переглянути результати будь якого користувача, а також змінити статус учня (рис. 6.11)

Рисунок 6.11 – Результати користувача

Висновки до розділу 6

В результаті було створено веб-додаток, у якого є всі функції для повноцінного навчання, та який за допомогою візуального моніторингу може чітко давати рекомендації стосовно того, як покращити свої результати. Також додаток має простий та інтуїтивно зрозумілий дизайн. Це було створено для того що б користувачі різних вікових груп, а також з різними навичками володіння комп’ютером, змогли швидко освоїти дану систему.

ВИСНОВКИ

В результаті виконання дипломної роботи була вирішена проблема з візуального моніторингу якості отримання знань. Задача є актуальною, оскільки онлайн курси і в загальному онлайн навчання, відзначається низьким рівнем завершеності та потворним вивченням навчального матеріалу. В процесі дослідження було розглянуто існуючі системи дистанційного навчання, та виявлено їхні недоліки.

На даний час в сучасному світі стрімко розвиваються і набувають популярності технології нейронних мереж. Зараз їх використовують майже всюди, в мобільних телефонах, додатках, веб-сайтах та в багатьох інших сферах. А також для розпізнавання людей, рухів, емоцій.

Проаналізувавши це було вирішено створити веб систему, до якої можливо буде підключити технологію з розпізнавання емоцій та концентрації уваги, для вирішення проблем дистанційного навчання. Система буде вести відео фіксацію користувача під час уроку та, після аналізу, надавати рекомендації що до покращення результатів.

Для демонстрації роботи системи було розроблено програмний продукт, що надає наступні можливості:

— проходження курсів користувачами з подальшим їх тестуванням;

— збереження відеоматеріалів та результатів тесту на сервері;

— можливість відправки відео на сторонній ресурс для розпізнавання емоцій;

— відображення результату виконання команд;

— можливість наповнення системи новим навчальним матеріалом.

Отже результатом роботи є система онлайн навчання з візуальним моніторингом якості отримання знань, яка є новим кроком для покращення дистанційного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brown, G.T.L. – 2017 - Assessment of Student Achievement – C. 10-48.

2. The State of Speech in HCI: Trend, Themes and Challenge // Interacting with Computer. – 2019. – №11. – С. 339–381.

3. Budhair, S.S. & Skipwith, K. // Best Practices in Engaging Online Learners Through – 2016 – С.–122–250.

4. Khare, A. & Hurst, D. On the Line - Business Education in the Digital Age.// Cham, Switzerland: Springerv – 2014 – С.20-30.

5. Aoun, J.E. Robot-Proof: Higher Education in the Age of Artificial Intelligence – 2018 –

6. Cooperman, L. The Art of Teaching Online: How to Start and How to Succeed as an Online Instructor – 2017 –. С.241.

7. Green, T.D. & Brown, A.H. The Educators Guide to Developing New Media and Open Education Resources – 2018 –.

8. Poritz, J.A. & Rees, J. Education is Not an App: The Future of University Teaching in the Internet Age. London: Routledge. – 2013 –. С.63-97.

9. Harasim, L. Learning Theory and Online Technologies. – 2017 –.

10. Andrew Krumm, director of learning analytics research at Digital//Learning Analytics Goes to School: A Collaborative Approach to Improving Education.2006

11. John Ebersole, former president (2006 to 2016) of Excelsior College, and William Patrick. Learning at the Speed of Light: How Online Education Got to Now.2015

12. Farhad Saba, professor emeritus of educational technology at San Diego State University, and Rick L. Shearer, director of research at Penn State World Campus. Transactional Distance and Adaptive Learning: Planning for the Future of Higher Education.

ДОДАТОК А

Інструментальні засоби системи з візуального моніторингу якості отримання занань

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_ АПЕПС_ТР61112_20Б

Аркушів 2

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР61112_20Б	Записка Мітряхін Ю.В.ТР-61.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР61112_20Б 12-1	entity	Компонент сутностей які відображаються в базі даних
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР61112_20Б 12-2	repository	Компоненти зв’язку з базою даних
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТР61112_20Б 12-3	controller	Компонент відображення веб сторінок

ДОДАТОК Б

Інструментальні засоби системи з візуального
Контролю якості отримання знань

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_ АПЕПС_ТР61112_20Б

Аркушів 16

Київ – 2020

```
@Controller
//Контролер головної сторінки веб системи
public class MainController {
    @Autowired
    private UserService userService;

    @Autowired
    private TopicsService topicsService;

    @Autowired
    private TegsService tegsService;

    @Autowired
    private CourseService courseService;


    //Метод який відповідає за відправку головної сторінки та її наповнення
    @RequestMapping ("/")
    public String getMainPage (Model model) {
        model.addAttribute ("easy", Difficulty.EASY);
        model.addAttribute ("medium", Difficulty.MEDIUM);
        model.addAttribute ("hard", Difficulty.HARD);
        model.addAttribute ("topics", topicsService.getAllTopics ());
        model.addAttribute ("tegs", tegsService.getAllTegs (). stream (). limit (10)
.collect (Collectors.toList ()));
        model.addAttribute ("courses", courseService.getAllCourse ());

        Authentication auth = SecurityContextHolder.getContext (). GetAuthentication
();
        model.addAttribute ("user", auth.getName ());
```

```
            model.addAttribute ("us", userService.findByUsername (auth.getName ()));

            if (auth! = null && auth.getAuthorities (). stream (). findFirst (). get () == Role.ADMIN) {
                model.addAttribute ("role", true);
            } else {
                model.addAttribute ("role", false);
            }

            return "main";
        }

    //Метод який дістає та відправляє всі теги з бази даних
        @ResponseBody
        @RequestMapping ("/ tags / get")
        public String getTegs () throws JSONException {
            return tegsService.convertToJson (tegsService.getAllTegs (). stream (). skip (10) .collect (Collectors.toList ()));
        }

    //Клас де зберігаються функції відповідні за функціонал тестів
    @Controller
    public class TestController {

        @Autowired
        private LessonService lessonService;
        @Autowired
        private TestsService testsService;
        @Autowired
        private ResultService resultService;
```

```
    @Autowired
    private UserService userService;

  //Функція який повертає сторінку з формою створення тесту
    @GetMapping (value = "/ test / add")
    @ResponseBody
    public ModelAndView editLesson (@RequestParam ("id") long id, Model model)
{
      model.addAttribute ("lesson", lessonService.findById (id));
      //model.addAttribute("type ", QuestionType.LEC);
      return new ModelAndView ("addTest");
    }
  \\ Функція яка отримує дані про тест та зберігає їх в базу даних
    @PostMapping (value = "/ test / save")
    @ResponseBody
    public int post (@RequestBody TestsDTO testsDTO) {
      System.out.println (testsDTO);
      testsService.save (testsDTO);
     / * for (int i = 0; i <testsDTO.getCorrect (). length; i ++) {
        for (int j = 0; j <testsDTO.getCorrect () [i] .length; j ++) {
          System.out.print (testsDTO.getCorrect () [i] [j] + "");
        }
        System.out.println ();
      } * /
      return 333;
    }
  //Функція яка отримує та зберігає результати тесту
    @PostMapping (value = "/ tests / passed")
    @ResponseBody
    public int passedTest (@RequestBody PassedTestDTO passedTestDTO) {
```

```
            Authentication auth = SecurityContextHolder.getContext (). GetAuthentication
();
            User user = userService.findByUsername (auth.getName ());
            System.out.println (passedTestDTO.toString ());
               try {
                  if (passedTestDTO.getQuestionId ()! = null) {
                     List <Question> mistake = testsService.checkTest (passedTestDTO);
                     resultService.saveResult (passedTestDTO, user, mistake, true);
                  } else {
                     resultService.saveResult (passedTestDTO, user, null, false);
                     return 202;
                  }
                  return 200;
               } catch (Exception e) {
                  return 333;
               }
            }

         //Функція що дістає результати тесту з бази та відправляє їх на сторінку
            @RequestMapping (value = "/ test / result")
            @ResponseBody
            public ModelAndView getResult (@RequestParam ("id") long id, Model model) {
               Authentication auth = SecurityContextHolder.getContext (). GetAuthentication
();
               User user = userService.findByUsername (auth.getName ());

                  if (user! = null && user.getAuthorities (). stream (). findFirst (). get () ==
Role.ADMIN) {
                     model.addAttribute ("role", true);
                  } else {
```

```
                model.addAttribute ("role", false);
            }
            model.addAttribute ("user", user.getUsername ());

        model.addAttribute ("result", resultService.findByLessonAndUser (user, lessonService.findById (id ())));
        return new ModelAndView ("resultPage");
    }

//Функція збереження відеофайлу
    @RequestMapping (value = "/ video / save", method = RequestMethod.POST)
    @ResponseBody
    public int saveVideo (@RequestParam ("video") MultipartFile file, boolean end) throws IOException {
        try {
            resultService.saveVideo (file, end);
            return 200;
        } catch (IOException e) {
            return 300;
        }
    }
//Видалення незакінченого відео матеріалу
    @RequestMapping (value = "/ video / delete", method = RequestMethod.GET)
    @ResponseBody
    public int deleteVideo () {
        System.out.println ("Delete reqwest");
      resultService.deleteVideo ();
      return 200;
    }
```

```
//Клас контролера  що відповідає за функціонал користувачів
@Controller
public class UsersController {

    @Autowired
    private UserService userService;

//Функція яка дістає та відправляє всі дані про користувачів
    @RequestMapping (value = "/ users / all")
    public String getAllUsers (Model model) {
        model.addAttribute ("users", userService.getAllUsers (). getUsers ());
        return "allUsers";
    }

//Функція відправки сторінки користувача з його інформацією
    @GetMapping (value = "/ user")
    @ResponseBody
    public ModelAndView getUsers (@RequestParam ("id") long id, Model model) {

        Authentication auth = SecurityContextHolder.getContext (). GetAuthentication
();
        User user = userService.findByUsername (auth.getName ());
        model.addAttribute ("logUser", user);
        model.addAttribute ("user", userService.findByUserId (id));

        return new ModelAndView ("userPage");
    }
    @GetMapping (value = "/ user / results")
    @ResponseBody
    public ModelAndView getUserResults (Model model) {
```

```
        Authentication auth = SecurityContextHolder.getContext (). GetAuthentication
();
        User user = userService.findByUsername (auth.getName ());
        model.addAttribute ("user", user);

        if (user! = null && user.getAuthorities (). stream (). findFirst (). get () ==
Role.ADMIN) {
            model.addAttribute ("role", true);
        } else {
            model.addAttribute ("role", false);
        }

        return new ModelAndView ("userResultPage");
    }
    //Функція яка по запиту змінює статус користувача
    @PostMapping (value = "/ user / change_type")
    @ResponseBody
    public int changeType (@RequestParam ("id") long id, @RequestBody boolean
type) {
        try {
            System.out.println ("User id -" + id + "; type -" + type);
            User user = userService.findByUserId (id);
            user.setPremium (type);
            userService.save (user);
            return 200;
        } catch (Exception e) {
            return 333;
        }
    }
```

```
//Клас контролера який відповідає за функціонал уроків
@Controller
public class LessonController {
    @Autowired
    private LessonService lessonService;
    @Autowired
    private CourseService courseService;
    @Autowired
    private TestsService testsService;
    @Autowired
    private QuestionService questionService;
//Функція яка відправляє сторінку додавання уроку
    @GetMapping (value = "/ lessons / add")
    @ResponseBody
    public ModelAndView deleteCourse (@RequestParam ("id") long id, Model model) throws DataSaveDBException {
        Course course = courseService.getCourseById (id);
        model.addAttribute ("Course", course);
        model.addAttribute ("Lessons", lessonService.findByCourseId (course));
        return new ModelAndView ("addNewLesson");
    }

//Функція яка отримує і зберігає дані про урок
    @SneakyThrows
    @RequestMapping (value = "/ lessons / save", method = RequestMethod.POST)
    public String saveTopic (@RequestParam ("files") MultipartFile [] files, LessonDTO topicDTO) {
        System.out.println (topicDTO.getCourseId () + "- course id");
        lessonService.saveLesson (topicDTO, files);
        return "redirect: /";
```

```
    }
//Функція яка видаля урок по ід
    @GetMapping (value = "/ lessons / delete")
    @ResponseBody
    public int deleteLesson (@RequestParam ("id") long id) {
        Lesson lesson = lessonService.findById (id);
        try {
            lessonService.delete (lesson);
            return 200;
        } catch (Exception e) {
            return 300;
        }
    }
//Функція яка відправляє сторінку редагування уроку
    @GetMapping (value = "/ lessons / edit")
    @ResponseBody
    public ModelAndView editLesson (@RequestParam ("id") long id, Model model)
{
        Lesson lesson = lessonService.findById (id);
        model.addAttribute ("lesson", lesson);
        model.addAttribute ("Lessons", lessonService.getAllLessons ());
        model.addAttribute ("Courses", courseService.getAllCourse ());
        return new ModelAndView ("editLesson");
    }
    @GetMapping (value = "/ lesson")
    @ResponseBody
    public ModelAndView lessonPage (@RequestParam ("id") long id, Model model)
{
        Lesson lesson = lessonService.findById (id);
```

```
            model.addAttribute ("lesson", lesson);
            model.addAttribute ("questions", questionService.findQuestionByTest
(lesson.getTest ()));
            model.addAttribute "" questionsSize ", questionService.findQuestionByTest
(lesson.getTest ()). size ());
            return new ModelAndView ("lessonPage");
        }
    //Клас контролер який відповідає за функціонал курсу
    @Controller
    public class CourseController {
        @Autowired
        public CourseService courseService;
        @Autowired
        public TopicsService topicsService;
        @Autowired
        public UserService userService;
        @Autowired
        public LessonService lessonService;
        @Autowired
        public TestsService testsService;
    //Функція яка повертає сторінку додавання курсу
        @RequestMapping ("/ courses / add")
        public String addBottle (Model model) throws DataSaveDBException {
            model.addAttribute ("easy", Difficulty.EASY);
            model.addAttribute ("medium", Difficulty.MEDIUM);
            model.addAttribute ("hard", Difficulty.HARD);
            model.addAttribute ("Topics", topicsService.getAllTopics ());
            return "addNewCourse";
        }
```

```
    //Функція яка отримує дані про новий курс та зберігає їх
        @SneakyThrows
        @RequestMapping (value = "/ courses / save", method = RequestMethod.POST)
        public String saveTopic (CourseDTO courseDTO) {
            System.out.println (courseDTO);
            courseService.saveCourse (courseDTO);
            return "redirect: /";
        }
    //Фукція яка по запиту повертає всі наявні курси
        @ResponseBody
        @RequestMapping ("/ courses / get")
        public String getCourses () throws JSONException {
            return courseService.convertToJson (courseService.getAllCourse ());
        }
    //Фнкція яка повертає сторінку курсу по ід
        @GetMapping (value = "/ course")
        @ResponseBody
        public ModelAndView getCourseById (@RequestParam ("id") long id, Model
model) throws IOException {
            System.out.println ("iiii" + id);
            Course course = courseService.getCourseById (id);

            Authentication auth = SecurityContextHolder.getContext (). GetAuthentication
();
            User user = userService.findByUsername (auth.getName ());

            if (course.isType () == false || (user! = null && course.isType () &&
user.isPremium ())) {
                if (user! = null && user.getAuthorities (). stream (). findFirst (). get () ==
Role.ADMIN) {
```

```
                model.addAttribute ("role", true);
            } else {
                model.addAttribute ("role", false);
            }

            model.addAttribute ("user", user.getUsername ());
            model.addAttribute ("course", course);
            model.addAttribute ("lessonsNum", course.getLessons (). size ());
            model.addAttribute ("lessons", course.getLessons ());
            return new ModelAndView ("coursePage");
        } else {
            return new ModelAndView ("noAccessToCourse");
        }

        // return "redirect: /";
    }
//Функція видалення курсу по ід
    @GetMapping (value = "/ course / delete")
    @ResponseBody
    public int deleteCourse (@RequestParam ("id") long id) {
        try {
            courseService.deleteCourse (id);
            return 200;
        } catch (Exception e) {
            return 300;
        }
    }
//Видалення теми курсу
    @GetMapping (value = "/ topic / delete")
    @ResponseBody
```

```
    public int deleteTopic (@RequestParam ("id") long id) {
        try {
            topicsService.deleteTopic (id);
            return 200;
        } catch (Exception e) {
            return 300;
        }
    }
//Функція яка повертає сторінку редагуваня курсу
    @GetMapping (value = "/ course / edit")
    @ResponseBody
    public ModelAndView editCourse (@RequestParam ("id") long id, Model model)
{
        model.addAttribute ("easy", Difficulty.EASY);
        model.addAttribute ("medium", Difficulty.MEDIUM);
        model.addAttribute ("hard", Difficulty.HARD);
        model.addAttribute ("Topics", topicsService.getAllTopics ());
        model.addAttribute ("course", courseService.getCourseById (id));
        return new ModelAndView ("editCourse");
    }

//Функція яка отримує назву нової теми та зберігає її
    @RequestMapping (value = "/ topic / add", method = RequestMethod.POST)
    @ResponseBody
    public int saveTopic (@RequestBody String topicName) {
        try {

            topicsService.save (topicName.replace ("\" "," "));
            return 200;
        } catch (Exception e) {
```

```
            return 300;
        }
    }
    //Контролер який відповідає за авторизацію користувачів
    @Controller
    public class AuthorizationController {
        @Autowired
        private UserService userService;
    //Функція яка повертає сторінку логіну
        @RequestMapping ("/ login")
        public String getLogin (@RequestParam (value = "error", required = false) String error,
                                @RequestParam (value = "logout", required = false) String logout,
                                Model model) {
            if (! SecurityContextHolder.getContext (). getAuthentication (). getName (). equals ("anonymousUser")) {
                return "redirect: / 403";
            }
            model.addAttribute ("error", error! = null);
            model.addAttribute ("logout", logout! = null);
            return "login";
        }
    //Фукція яка повертає сторінку реєстрації
        @PostMapping (value = "/ register")
        public String getLogin (UserDTO userDTO) {
            try {
                userService.saveNewUser (userDTO);
            } catch (DataSaveDBException e) {
                return "redirect: / registration? error";
            }
```

```
        return "redirect: /";
    }
//Функція яка зберігає введенні дані нового користувача
    @RequestMapping ("/ registration")
    public String getLogin (@RequestParam (value = "error", required = false) String
error,
                            Model model) {
        model.addAttribute ("error", error! = null);
        return "registration";
    }
```

ДОДАТОК В

Інструментальні засоби системи з візуального контролю якості отримання знань

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_ АПЕПС_ТР61112_20Б

Аркушів 10

Київ – 2020

АНОТАЦІЯ

Додаток містить опис основних програмних класів інструментальних засобів веб-системи з моніторингу якості отримання знань, що виконують деякі головні функціональні можливості програми, а саме:

— Можливість відображення веб сторінок в залежності від посилання.

— Можливість зв’язку з базою даних за допомогою сервісів та фреймворку Hobernate.

Веб-система розроблена за допомогою мови програмування Java, фреймворку Spring, для клієнтської частини використано JavaScript, HTML,CSS

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ 69
2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ 70
3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ 71
4. ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУЮТЬСЯ 72
5. ВИКЛИК І ЗАВАНТАЖЕННЯ 73
6. ВХІДНІ ДАНІ 74
7. ВИХІДНІ ДАНІ 75

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис основних програмних класів інструментальних засобів системи з візуального контролю якості отримання знань, що виконують деякі головні функціональні можливості програми.

У додатку Б міститься програмний код основних програмних модулів таких як контролерів.

Для роботи даної системи потрібно мати пристрій з можливістю виходу в глобальну мережу за допомогою одного із браузерів: Google Chrome, Mozilla Firefox, Opera. А також стабільне інтернет з’єднання, не повільніше ніж 5мб/с.

Веб-система розроблена за допомогою мови програмування Java, фреймворку Spring, для клієнтської частини використано JavaScript, HTML,CSS.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Веб-система з візуального моніторингу якості отримання знань, надає такі функціональні можливості:

1. Можливість авторизації та реєстрації.
2. Можливість обрати мову на які буде відображатись сайт
3. Можливість пошуку курсу по критерієм
4. Можливість проходження уроку та тестування
5. Відеофіксація користувача під час уроку
6. Можливість надання результатів та порад по закінченню тестування
7. Можливість додавання, редагування, видалення курсів та уроків викладачами
8. Можливість перегляду викладачами результатів користувача.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою, додаток складається з трьох головних модулів:

— шар представлення.

— шар контролерів.

— шар моделі.

Шар представлення містить сторінки які відображаються на стороні користувача.

Шар моделей містить в собі класи, що використовуються для моделювання сутностей, якими оперує веб-система.

Шар контролерів пов’язує між собою логіку яка міститься в моделі та представлення у якому відображається інтерфейс.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання веб системи користувач повинен мати пристрій з підтримкою сучасних браузерів, а також стабільне мережеве з’єднаня, зі швидкістю не менше як 5мб/с.

Для першого використання застосунку користувач має мати підключення до мережі інтернет для авторизації у систему. Надалі у користувача є доступ до всіх безкоштовних курсів. Що б отримати доступ до преміум курсів користувачу потрібно звернутись до викладача.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для завантаження веб-системи в локальній системі потрібно встановити базу даних MySQL та в property проекту вказати дані sql сервера. Після запуску програми потрібно перейти за посиланням localhost:8080/.

Для того щоб система стала доступна всім користувачам, її потрібно інсталювати на одній із хостингових платформ, вибравши доменне ім’я сайту.

ВХІДНІ ДАНІ

Вхідна інформація для інструментальних засобів веб застосунку:

1. Вхідними даними системи аутентифікації мають бути логін та пароль.
2. Вхідними даними системи реєстрації мають бути електронна пошта, оригінальний логін та пароль.
3. Вхідними даними курсу має бути ім’я, тема, опис, тип та тегі.
4. Вхідними даними уроку має бути ім’я, опис, текст уроку, фотографії та посилання з YouTube.
5. Вхідними даними тесту має бути питання, вибір типу питання, та варіанти відповідей з вибором правельних.

ВИХІДНІ ДАНІ

Вихідними даними системи з візуального моніторингу якості отримання знань є структуроване відображення інформації курсів та уроків. Також проаналізовані результати тестів користувача з порадами про те як покращити результат.